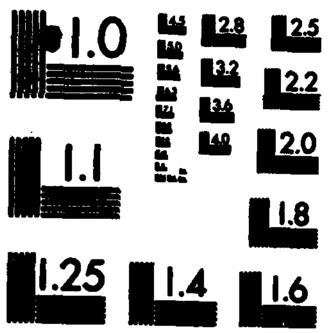
AD-A146 392 INVESTIGATIVE FINDINGS OF IMAGE ENHANCEMENT TECHNIQUES 1/2
(U) JAYCOR ALEXANDRIA VA W B LANDER 26 MAR 84
JAYCOR-J206-84-004/6238 SBI-AD-E001 736
UNCLASSIFIED N00014-83-C-2273 F/G 17/8 NL



'OPY RESOLUTION TEST CHART

2

ADE 001736

AD-A146 392

INVESTIGATIVE FINDINGS OF IMAGE ENHANCEMENT TECHNIQUES

J206-84-004/6238

JAYCOR

560,659

205 South Whiting Street
Alexandria, Virginia 22304

JAYCOR

2

INVESTIGATIVE FINDINGS OF IMAGE ENHANCEMENT TECHNIQUES

J206-84-004/6238

Final Report

by

W. Brent Lander

March 26, 1984

Prepared for:

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375

Under:

Contract Number N00014-83-C-2273

JAYCOR

6238

March 26, 1984

Dr. R. A. Steinberg
Code 6509
Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375

SUBJECT: JAYCOR Final Report, Contract Number N00014-83-C-2273

Dear Dr. Steinberg:

JAYCOR is pleased to submit this Final Report entitled, *Investigative Findings of Image Enhancement Techniques*, in accordance with the subject contract, CDRL Item Number A002.

If the Final Report is acceptable, please sign and forward the enclosed DD Form 250.

Questions of a technical nature should be addressed to Mr. W. Brent Lander while questions of a contractual nature should be addressed to Mr. Floyd C. Stilley, our Contracts Administrator.

Sincerely,

[illegible]

Martin C. Nielsen
Vice President of Administration
and Counsel

ssh-b

Enclosures

cc: Code 6502.2
Code 2627
DTIC

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER J206-84-004/6238	2. GOVT ACCESSION NO. ADA146392	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INVESTIGATIVE FINDINGS OF IMAGE ENHANCEMENT TECHNIQUES		5. TYPE OF REPORT & PERIOD COVERED Final Report: 08/17/83 thru 01/16/84
		6. PERFORMING ORG. REPORT NUMBER J206-84-004/6238
7. AUTHOR(s) W. Brent Lander		8. CONTRACT OR GRANT NUMBER(s) N00014-83-C-2273
9. PERFORMING ORGANIZATION NAME AND ADDRESS JAYCOR 205 South Whiting Street Alexandria, VA 22304		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS A002
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Research Laboratory 4555 Overlook Avenue, SW Washington, DC 20375		12. REPORT DATE March 26, 1984
		13. NUMBER OF PAGES 123 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

2 copies - Code 6509
1 copy - Code 6502.2
6 copies - Code 2627
12 copies - DTIC

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION IS UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

FILTER

Interactive Image Filtering Programs

Interactive Image Filtering Program

by

William Brent Lander

JAYCOR

1. FILTER

FILTER is a program to do interactive image filtering using a simple algebraic notation. Commands are of two forms.

Form 1:

```
filter> DESTINATION = FILTER EXPRESSION
```

Form 2: Program Control

```
filter> COMMAND OPERANDS
```

Where COMMAND is one of the commands described in the section 3.

1.1 FILTER Vita

FILTER was developed at the Naval Research Laboratory Code 6520 Advanced Concepts Branch Computer Facility. The computer code was written and debugged on the VAX 11/780 computer with the De Anza/Gould IP8500 image processor. FILTER consists of approximately 4500 lines of FORTRAN Four Plus, as provided by Digital Equipment Corporation, and shares approximately 2300 lines of code with the MATH program, also, many operating system services were used. The program was developed under versions 3.2, 3.3, and 3.4 of VMS. Level 0 (Device Driver) of the De Anza/Gould LIPS 3.0 package was used, although all other image processor control functions are done through custom written software.

The code was designed and developed by William Brent Lander. Debugging has been done while in use for several months at NRL.

1.2 The Internal Blackboard

Many filters require various parameters which are almost always the same, but occasionally must be changed. Once these parameters are changed, the alteration must stay in effect for a number of operations. For example, most operations will be done on an entire image, however sometimes it is desired to modify parameters to do work on only a fraction of the overall image. Another example is the setting of neighborhood sizes for either equalization or some edge detectors.

To handle this situation, FILTER has built in to it a working memory which is used to store values for the filtering parameters. When a filter needs a parameter, a request is made to obtain values from the blackboard. If a value is found, that value is returned. If no value is found, the blackboard manager then asks the terminal for a value for that parameter. When a value is entered from the terminal, the blackboard manager both returns the value to the program that requested the information, as well as entering the data in the blackboard for future reference.

To display the contents of the blackboard, or to place a value into the blackboard, FILTER offers the SET and SHOW commands. These commands are described in the section of this report on Form 2 commands.

2. Form 1 Commands

Form 1 commands are the active commands to do image filtering. The basic form of the FILTER assignment command is as follows

```
filter> {destination} = {filter} {expression}
```

As with the MATH program, {destination} can be either

1) A number from 0 to 20. This will cause the output to go to that Image Refresh Memory (IRM)

2) Filename. This will cause the output to go to the requested file. New files are created in standard image format.

An {expression} is

1) A filename. The contents of the file (in standard format) is to be acted upon.

2) A numeric constant. Sets the input to the filter to be a constant.

3) A string of the form *IRMn* where n is an integer in the range 0 to 20. This causes the input to be from a given IRM.

4) An equation, in infix notation, which may combine items from 1, 2, or 3.

Parenthesis can be used to specify order of evaluation.

2.1 Examples

The following example does a uniform equalization on the ratio of two files: WASH.B1 and WASH.B6. The result of the equalization is sent to image refresh memory 0.

```
filter> 0=uniform Wash.B1/Wash.B6
```

Both upper and lower case letters can be intermixed. Internal to FILTER, all input is translated to uppercase before the command is decoded.

A second example is shown by applying the KIRSCH edge detector to the difference between two files. Here the output is sent to the disk file DIFF.PIX

```
filter> Diff.Pix = KIRSCH Abs(File.B1 - File.B2)
```

In the expression to be filtered, blanks can be used at any time, with the exceptions of 1) filenames cannot contain blanks imbedded in them, and 2) the image refresh memory identifiers (i.e. IRM0, IRM1, IRM2 . . .)

A documented list of available filters is shown in section 4.

For a more complete discussion of possible image expressions, see the documentation on the MATH program.

3. Form 2 Commands

3.1 END

```
filter>END
```

Can be used to cause the FILTER program to terminate. FILTER can also be exited by entering EXIT, or Control-Z.

3.2 HELP

FILTER has an on-line help facility that can be used interactively to obtain information about the FILTER commands, the individual filters, or the parameters from the blackboard. Approximately fifteen hundred lines of documentation are available.
To use the help facility type HELP in response to the filter prompt:

```
filter> HELP
```

A brief introduction to FILTER will be typed, followed by a listing of available subtopics. By typing the name of a subtopic, you can progress to the next level of help. By typing in a blank line, you can move up one level. By typing in Control-Z, you can return to the main level filter prompt.

3.3 OUTPUT

```
filter> OUTPUT <ascending|descending>
               <flipped|notflipped>
               <vertical|horizontal>
               <?>
```

Controls the order of storage of the output. Ascending|Descending and Flipped|Notflipped apply to both file and image output. Vertical|Horizontal applies only to image output and not to the file output. The "?" will cause the current setting to be displayed.

For example, to specify that the output should be across the screen from left to right with the image line-wise reversed, the following command would be used:

```
filter> Output Horizontal Flipped
```

3.3.1 ASCENDING

Causes the output to start with the first record in the file/ line 0 of the display, and output to sequentially higher numbered records.

3.3.2 DESCENDING

Causes the output to start with the last record in the file/line 511 of the display, and output to sequentially lower numbered records.

3.3.3 FLIPPED

Causes the output to be transposed on a line by line basis before output. The first pixel is exchanged with the last pixel on the line, the second pixel is exchanged with the next to last pixel, and so on.

3.3.4 HORIZONTAL

Causes the output to be painted to the screen from left to right, or right to left, as opposed to top to bottom.

3.3.5 NOTFLIPPED

Causes the output to be in the same record for record order as input. The first pixel in the image file is the first pixel in the output.

3.3.6 VERTICAL

Causes the output to be painted to the screen from top to bottom, or bottom to top.

3.4 SET

Is used to set blackboard values to a constant. Syntax is as follows:

```
filter> SET <variable> <value>
```

3.4.1 TRACE

If the value of the variable TRACE is set to 1, every time a value is obtained from the blackboard, the variable, its value, and where the values is requested from is typed at the terminal.

3.5 SHOW

Display blackboard value at the terminal.

```
filter> SHOW <variable>
```

3.6 $ - Executing a DCL Command

```
filter>$<DCL-COMMAND>
```

Causes the filter program to "go to sleep" and runs the requested DCL command. This can be very useful in cases where the name of the picture file has been forgotten and a directory command would be beneficial.

3.7 @ - Command_File

```
filter>@<FILTER-COMMAND-FILE>
```

Causes the filter program to run with input from a file, as opposed to the terminal. Useful when the same set of operations will be used over and over. Any filter command is valid in a command file.

3.8 ! - Comment

If the first non-blank symbol is an exclamation mark(!) the line is ignored. Comment lines are especially useful either in command files or in producing examples of terminal sessions.

Example:

```
filter> !The following code shows the Sobel edge
filter> ! operator in action.
filter> !
filter> O=SOBEL Temp.pic
filter> ! . . .
```

4. Available Filters

For this report, a filter is any one of the image processing routines that are available in the FILTER program. Processing is available in five areas at this time: Noise reduction, Equalization, Edge detection, Noise introduction and Statistical measurement.

Noise reduction routines include AVERAGE, MEDIAN, PRATTM1, PRATTM2, and PRATTM3. These routines return data which has been processed to remove random pixel type of noise.
Equalization routines include EXPONENTIAL, RAYLEIGH, and UNIFORM. These routines map the intensities in the input into the same range, but over a different distribution. In this way, the data can often be better displayed.

Edge detection routines, such as the CROSS, SOBEL, and KIRSCH, are useful in bringing out the structure of image data. Texture studies can also be made using these routines.

Noise introduction routines can be used to study how various image processing techniques behave in a less than ideal surroundings. These routines allow for the user to build a noisy environment to model how some of the data occurs.

Statistical Measurement routines provide useful information on the shape of the intensity histogram of the input expression. Typical of these routines are the more standard MEAN, VARIANCE, RMS, and STANDARD_DEVIATION. A number of the more exotic measures have been included, such as ENERGY, ENTROPY, SKEW, and KURTOSIS. These more exotic routines tend to take more processing time, so the user should be wary of over using them.

All filters use the blackboard to determine the size of the field they will be operating on in the overall 512 by 512 image. The blackboard parameters FIRSTROW, FIRSTCOL, LASTROW, and LASTCOL define the region over which filtering will be done. These values are initially set so that processing will be done over the entire image.

4.1 ARGYLE

Argyle is a linear edge detector useful for enhancing contrast and bringing out edges. Argyle operates over a rectangular mask of size ROWMASK pixels by COLMASK pixels. The mask contains an approximation of a Gaussian spread function defined by ROWSPREAD and COLSPREAD. Typical values for ROWSPREAD and COLSPREAD range from 3 through 20. ROWMASK and COLMASK are set to 5 by default, but can range up through 20. For more information see E. Argyle; "Techniques for Edge Detection", Proc. IEEE, 59, 2, Feb. 1971, Pages 285-287.

4.2 AVERAGE

AVERAGE will blur the input expression, each pixel receiving the average value over a user defined rectangle of pixels. In this way the picture is lowered in contrast where point irregularities are reduced because of the image intensity around them. MEAN is a synonym for AVERAGE. The averaging is done over a neighborhood of dimension ROWWINDOW by COLWINDOW.

4.3 BLOCK

BLOCK divides the image into a number of rectangles and replaces the value of the pixels in the rectangle with the average value in the rectangle. As with the AVERAGE filter, the rectangles are of size ROWWINDOW by COLWINDOW.

4.4 CROSS

Robert's Cross edge detector algorithm. This is a 2 by 2 pixel, non-linear edge detector. The small size of the detector makes it very useful in doing analysis of data with small features or sharp edges.

4.5 ENERGY

The ENERGY filter gives a pixel value of the value of the statistical energy of the ROWWINDOW by COLWINDOW neighborhood around the pixel in the input expression. The energy is defined as the sum of the squares of the probability of each intensity. The energy value is then multiplied by 256 to fill the normal intensity range. For a reference on energy measures in pictures, see William Pratt, "Digital Image Processing", Wiley-Interscience, 1978, pages 471-474.

Please Note: The energy is a difficult filtering function and can take over an hour to do a simple 512x512 picture. Processing times will vary due to the sizes of ROWWINDOW and COLWINDOW. In general, larger windows take longer.

4.6 ENTROPY

The ENTROPY filter gives a pixel value of the value of the statistical entropy of the ROWWINDOW by COLWINDOW neighborhood around the pixel in the input expression. The entropy is defined as the negative of the sum of the product of the probability of each intensity times the log(base 2) of that same probability. The entropy value is then multiplied by 256 to fill the normal intensity range. For a reference on entropy measures in pictures, see William Pratt, "Digital Image Processing", Wiley-Interscience, 1978, pages 471-474.

Please Note: The entropy is a difficult filtering function and can take over an hour to do a simple 512x512 picture. Processing times will vary due to the sizes of ROWWINDOW and COLWINDOW. In general, larger windows take longer.

4.7 EXPONENTIAL

Exponential histogram equalization on a block by block basis. The size of the blocks are determined by the values of ROWEQUAL and COLEQUAL. The equalization constant is set by E_ALPHA. Typical values are from -1. to 1.

4.8 GAUSSIAN

Adds normally distributed noise onto the input image. The distribution of the noise can be controlled by the MEAN and STAND.DEV blackboard parameters.

4.9 HYSTERSIS

Hystersis smoothing on all or part of an image. Hystersis smoothing is a non-linear method of reducing small variations in images due to "shakey low-order bits". This method minimally blurs the picture. For a more complete description of how this algorithm works, see Duda and Hart, "Pattern Classification and Scene Analysis", 1973.

The magnitude of the blackboard parameter NUMHYST controls the number of times vertical or horizontal smoothing is done. If NUM_HYST is less than 0, smoothing is first done in the vertical direction (column-wise), otherwise smoothing is first done in the horizontal direction (row-wise).

The Blackboard parameter HYSTERSIS controls the size of the cursor window used.

4.10 KIRSCH

The Kirsch edge detector is a non-linear three by three edge operator. Eight directional differences are measured, and the edge is assigned the greatest of the differences. For more information, see R. Kirsch, "Computer Determination of the Constituent Structure of Biological Images", Computers and Biomedical Research, 4,3, 1971, pages 315-328.

4.11 KURTOSIS

The KURTOSIS filter gives a pixel value of the value of the skewness in histogram of the ROWWINDOW by COLWINDOW neighborhood around the pixel in the input expression. The kurtosis is defined as the sum of the fourth power of differences between the individual intensities and the mean intensity where the sum is divided by the square of the variance of the points. The kurtosis value is then multiplied by 100 to fill the normal intensity range. In this scale, a zero mean normal distribution will have a value of 100. For a reference on energy measures in pictures, see William Pratt, "Digital Image Processing", Wiley-Interscience, 1978, pages 471-474

Please Note: The Kurtosis is a difficult filtering function and can take over an hour to do a simple 512x512 picture. Processing times will vary due to the sizes of ROWWINDOW and COLWINDOW. In general, larger windows take longer.

4.12 LOGRITHMETIC

Block logrithmetic equalization filter. This filter performs the logrithmetic equalization over each block of size ROWEQUAL by COLEQUAL. The product of ROWEQUAL and COLEQUAL should be greater than 30 and closer to 300 for best results.

4.13 MACLEOD

Linear edge detector which convolves a variable sized rectangle of dimension ROWMASK by COLMASK which contains Gaussian edge masks defined by ROWSPREAD and COLSPREAD to the input expression.

4.14 MAXIMUM

MAXIMUM is a filter which replaces the value of each pixel by the maximum value of the pixels in the surrounding rectangular neighborhood defined by ROWWINDOW by COLWINDOW.

4.15 MEAN

See documentation on AVERAGE or BLOCK.

4.16 MEDIAN

Does simple median filtering over a series of rectangular image segments. The value of each pixel is replaced by the median of the pixel values in a rectangular region around that pixel. The neighborhood is of size ROWWINDOW by COLWINDOW.

4.17 MINIMUM

Minimum is a filter which replaces the value of each pixel by the minimum value of the pixels in the surrounding rectangular neighborhood defined by ROWWINDOW by COLWINDOW.

4.18 PRATTM1

This is a linear, convolution mask type of noise cleaning algorithm. The mask is three by three with weights as follows

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

For more information, see "Digital Image Processing" by William Pratt, 1978, pages 319-321.

4.19 PRATTM2

This is a convolution mask type of noise cleaning algorithm. The mask is three by three with weights as follows:

1/10	1/10	1/10

1/10	1/5	1/10
1/10	1/10	1/10

For more information, see "Digital Image Processing" by William Pratt, 1978, pages 319-321.

4.20 PRATTM3

This is a linear, convolution mask type of noise cleaning algorithm. The mask is three by three with weights as follows:

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

For more information, see "Digital Image Processing" by William Pratt, 1978, pages 319-321.

4.21 RAYLEIGH

Rayleigh histogram equalization on a block by block basis. The size of the blocks are determined by the values of ROWEQUAL and COLEQUAL. The equalization constant is set by R_ALPHA. For more information on the RAYLEIGH equalization, see W.Pratt "Digital Image Processing", 1978.

4.22 RMS

The RMS filter gives a pixel value of the value of the root-mean-square of the ROWWINDOW by COLWINDOW neighborhood around the pixel in the input expression.

4.23 SKEW

The SKEW filter gives a pixel value of the value of the skewness in histogram of the ROWWINDOW by COLWINDOW neighborhood around the pixel in the input expression. The skew is defined as the sum of the cube of differences between the individual intensities and the mean intensity where the sum is divided by the cube of the standard deviation of the points. The skew value is then multiplied by 256 to fill the normal intensity range. For a reference on energy measures in pictures, see William Pratt, "Digital Image Processing", Wiley-Interscience, 1978, pages 471-474.

Please Note: The skew is a difficult filtering function and can take over an hour to do a simple 512x512 picture. Processing times will vary due to the sizes of ROWWINDOW and COLWINDOW. In general, larger windows take longer.

4.24 SOBEL

Performs the Sobel edge operator over the input image. The Sobel operator is a three by three operator which replaces the central element with the greater of the difference between the top and bottom rows, or the right and left columns.

4.25 STANDARD_DEVIATION

The STANDARD_DEVIATION filter gives a pixel value of the value of the square root of the variance for the ROWWINDOW by COLWINDOW

4.26 STAT_DEF

The statistical edge detector is a non-linear ROWWINDOW by COLWINDOW operator which produces as output the ratio of the value of the pixel to the standard deviation of the pixels in the rectangular region around it.

4.27 UNOISE

Add UNIFORM noise onto the input image. The noise added is uniformly distributed between the values LOWERNOISE and UPPERNOISE. If

```
LOWERNOISE == -UPPERNOISE
```

the overall intensity of the picture will not change.

4.28 UNIFORM

Block uniform equalization filter. This filter stretches the intensity histogram equalization over each block of size ROWEQUAL by COLEQUAL to the range 0 to 255.

4.29 VARIANCE

The VARIANCE filter gives a pixel value of the value of the variance of the ROWWINDOW by COLWINDOW neighborhood around the pixel in the input expression.

4.30 WALLIS

The Wallis edge detector is a non-linear three by three edge operator. The value of each pixel is determined to be the logarithm of the ratio of the fourth power of the original pixel intensity, with the product of the neighboring pixels.

IIU

Interactive Image Utility Programs

Interactive Image Utility Package

by

William Brent Lander
JAYCOR

1. Introduction

The routines described in this manual are available as system commands on the NRL Code 6520 VAX 11/780 computer. These programs allow for the novice digital image processing to do complex and meaningful operations on real-world data.

Programs are provided for data display, intensity stretching, image labeling, interactive graphics drawing, contour and surface plotting, and handling of LANDSAT images.

1.1 Vita

These programs were developed at the Naval Research Laboratory,Code 6520, VAX 11/780 computer center. The computer has 2.0 megabytes of MOS memory and 8.0 megabytes of virtual memory. The computer has attached to the Unibus a Gould/DeAnza IP8500 image processor.

Programs documented here were all written in VMS Fortran Four Plus as provided by DEC. Extensive use of operating system services is made.Also, these programs use level 0 (Device Driver) of the DeAnza LIPS software. All more advanced routines are provided by the IP8 subroutine package.

All code was designed and implemented by William Brent Lander.

1.2 Accessing Programs in the IIU package

All of the programs in the IIU package, except the demonstration programs, have been installed as foreign commands on the VAX 11/780 belonging to the Naval Research Laboratories Code 6520 Advanced Concepts Branch. Because of this, it is only necessary to enter the name of the program desired in order to get that program to run on the terminal.

The programs were designed to require minimum key strokes. For that reason, in most cases, it is only necessary to enter one symbol to cause the program to execute that command(e.g. pushing <return> is not necessary). The only exception to this rule is when a command needs a qualifier, such a numeric constant. In those cases, it is necessary to end the command by typing <return>. The individual IIU programs will state which commands will require additional input.

The programs may be exited by any of the following techniques:

```
1) E .........Entering the E for Exit routine
2) <return>...Striking the RETURN key with nothing on
                  the line
3) <control-Z>.Striking the letter Z while holding
                  down the control key.
```

1.3 Using the Joystick

Many of the routines described in this document make use of the DeAnza joystick box. Physically, this is a sloping paneled box, about eight by six inches on the base and four inches tall. In the center of the box top, there is a polished chrome control lever, and on the left hand side there is a column of five switches. The switches control the mode of operation of the joystick, while the degree which the chrome control lever is moved to one side controls the graphics information.

The joystick usually works in tandem with the dual cursor generator in the DeAnza. Display cursor one is controlled by the joystick when the switch at the top of the joystick box is turned to the ON position. Display cursor two is controlled by the joystick when the second switch from the top of the joystick is turned to the ON position. When both switches are turned on, the two cursors are controlled together. When only one switch is turned on, one cursor can be moved with respect to the other. When both cursors are off, the position of the chrome box has no effect on the cursor positioning.

When a box is displayed on the monitor, this means that both cursors are active. The box displayed can be moved by placing the first two switches in the ON position and moving the chrome switch. If only one switch or the other is ON, moving the chrome switch will cause three corners of the box to move, and one to stay constant. In this way the shape of the box can be changed.

The next two switches on the joystick box are labeled TRACK and MODE. Both of these switches should remain off for use with all of the programs in this document.

The final switch is the ENTER button. It is not used by any programs in the IIU package at this time.

2. BITS

BITS does bit manipulations on whole images displayed on the DeAnza image processor. With BITS, it is possible to produce displays where the image reflects the lower-order bits of display data. The transforms possible are designed to look for data errors which are the result of digitization problems. For example, one digitization problem that can occur is sticky low-order bits from the analog to digital converter circuits. BITS can be used to bitwise rotate through the pixel bytes in the display so that when this problem happens, the display will show line segments along the direction of the data scanning.

2.1 Commands

Commands are of the form:

```
0.......Use IRM 0 for output
1.......Use IRM 1 for output
2.......Use IRM 2 for output
3.......Use IRM 3 for output
A.......Absolute data. Show with no modifications.
Fx......Fix data.  Not available at this time.
E.......End program also <space> will terminate
        program.
H.......On-Line help
Mx......Mask  data (MH for on-line help)
Rx......Rotate data (RH for on-line help)
Sx......Shift  data (SH for on-line help)
X.......Bit reversal (bit 8 goes to bit 1,
        bit 7 to bit 2...)
$.......Execute DCL command
@.......Do BITS command file
!.......Comment line.
. ......Toggle on/off terminal output
```

2.2 Mask Operations

Mask operations are designed to do a logical AND operation of the bits on the display with the value defined in the mask. Only those bits which are turned on in the mask operand, and those turned on in the pixels will be shown.

The valid mask operands are as follows:

```
MDn.....Mask decimal, where n is the mask in decimal
```

MH......Evoke on-line help
MOn.....Mask octal, where n is the mask in octal
MXn.....Mask hexidecimal, where n is the mask in
hexidecimal

2.3 Rotate Operations

Rotate operations take the bit patterns in the pixels and shift the bits right or left as requested. Those bits which are shifted out of bounds are tacked on to the opposite side of the pixel.

Valid rotate operations are:

RH......Evoke on-line help
RLn.....Rotate Left by n bits
RRn.....Rotate Right by n bits

2.4 Shift Operations

Shift operations take the bit patterns in the pixels and do logical shifts of the bits as requested. Bits that are shifted outside of the byte range are not displayed. As with all BITS operations, the effects visualized on the screen are produced by loading transformation tables, so no real image data is lost.

Valid shift operations are as follows:

SH......Evoke on-line help
SLn.....Shift Left by n bits
SRn.....Shift Right by n bits

NOTE: All bit manipulations supersede each other

Normally, operations are done the moment the keystroke is complete, however on those operations requiring numeric input, a carriage return is necessary to terminate the numeric input.

2.5 Command Files

BITS does allow for command files. In command files, all cursor input is replaced by numeric input on the line of the command. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the BITS commands.

2.6 Notes

1) BITS was designed to look for subtile variations in bit patterns produced by faulty A/D converters for images. When the types of operations that BITS provides are used on suspect data, low-order bits which are stuck on or off will cause irregular features to show up.

2) BITS uses the lookup tables in the hardware to alter the display. As such, the operations are done extremely fast, and no permanent damage is done to the data.

3) This program can be used to look at data in all three image refresh memories, although the channels are handled separately, the data manipulations are in effect after the channel to examine has been changed.

3. CLOSEUP

CLOSEUP uses the hardware zoom and scroll registers to allow the user to examine detail in images. The cursor box is used to point to a region of interest on the screen. By pressing the plus (+) or minus (-) keys the magnification of the display can be changed. Eight levels of magnification are available.

The magnification is simple linear expansion. When a region is expanded, the area appears to be filled with little boxes. Magnification is done at the 1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 1:7, and 1:8 levels.

3.1 Commands

Commands are of the form:

```
C.......Center the picture to fill the frame
E.......End program also <space> will terminate
        program.
H.......On-Line help
+ ......Increase Zoom
- ......Decrease Zoom
$.......Execute DCL command
@.......Do CLOSEUP command file
!.......Comment line.
. ......Toggle on/off terminal output
```

3.2 Command Files

CLOSEUP does allow for command files. In command files, all cursor input is replaced by numeric input on the line of the command. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the CLOSEUP commands.

3.3 Notes

1) Most programs that use the hardware cursors assume that the picture is unzoomed, (i.e. at the 1:1 magnification level). For that reason, other operations, such as running HISTO, should be done before running CLOSEUP.

4. CONTOUR

Contour produces iso-intensity contour graphs of images displayed on the DeAnza image processor. Both the original image, and the output contour are output to the DeAnza. The graphics produced use the overlay graphics planes and do not affect the data being displayed.

4.1 Commands

Commands are of the form:

```
0.......Sample IRM 0 for plot(default)
1.......Sample IRM 1 for plot
2.......Sample IRM 2 for plot
3.......Sample IRM 3 for plot
A.......Automatically set display bounds to
        data(default)
B.......Draw plot in blue
C.......Produce contour plot
G.......Draw plot in green(default)
E.......End program
H.......Help
K.......Kill (clear) graphics  on screen
Ln......Set lower bound on intensities displayed
Ox......Output is to be drawn to region
R.......Draw plot in red
Un......Set upper bound on intensities displayed
Vn......Draw contour of value n
W.......Use cursors to get windowed area of picture
$.......Execute DCL command
@.......Do CONTOUR command file
!.......Comment line
. ......Toggle on/off terminal output
```

Normally, operations are done the moment the keystroke is complete, however on those operations requiring numeric input (e.g. where the command letter is followed by the letter n), a carriage return is necessary to terminate the numeric input.

4.2 Output Commands

The image produced by CONTOUR can be displayed in all or part of the screen. The destination of the output is controlled by the Output command. Valid output commands are as follows:

```
OA........Output to entire screen
OH........Produce this listing
OC........Output to region bounded by cursor
```

4.3 Command Files

CONTOUR does allow for command files. In command files, all cursor input is replaced by numeric input on the line of the command. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the CONTOUR commands.

4.4 Notes

1) The cursor positioning in CONTOUR assumes that the picture is neither scrolled nor zoomed. For this reason, it is necessary to run CLOSEUP on pictures after CONTOUR is run, rather than using CLOSEUP before CONTOUR.

2) CONTOURing takes time. For this reason, it is best to keep the region under examination small.

4.5 Example

Normally, to produce a contour plot of a portion of a picture the person evokes the contour plotting program, uses the W command to select a portion of the picture to analyze, sets the program to draw the contour plot in the region over the selected region (the OC<return> command),specifies an iso-intensity line to be drawn (the V command), and issues the command to do the plot (the C command). The terminal session to do this would appear as follows:

```
$CONTOUR
contour>W
Move the cursor box to the position
requested, and press <return> when ready
.... the cursor is positioned for size and
         location here...

contour>OC<return>
contour>V100<return>
contour>C
```

5. CONVOLVE

CONVOLVE performs NxM convolutions on single or multiple Image Refresh Memory (IRM) images displayed on the DeAnza image processor. The images are assumed to already be available on the screen.

5.1 Commands

Commands are of the form:

```
A.......Convolve on all of display(default)
C.......Do convolution
E.......End program.
G.......Graphically enter convolution weights
H.......Evoke On-Line help
Mn......Set number of Columns in convolution
        (default 3)
Nn......Set number of Rows in convolution(default 3)
On......Define output Irm for convolution
        (default IRM 1)
S.......Scale output to display limits
T.......Truncate output to display limits(default)
V.......View convolution graphically
W.......Use cursors to window region on which to
        operate
$x......Execute DCL command
@x......Do CONVOLVE command file
!x......Comment line.
. ......Toggle on/off terminal output
```

Normally, operations are done the moment the keystroke is complete, however on those operations requiring numeric input(e.g. those having the letter n after the command letter), a carriage return is necessary to terminate the numeric input.

5.2 Command Files

CONVOLVE does allow for command files. In command files, all cursor input is replaced by numeric input on the line of the command. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the CONVOLVE commands.

5.3 Notes

1)The cursor positioning in CONVOLVE assumes that the picture is neither scrolled nor zoomed. For this reason, it is necessary to run CLOSEUP on pictures after CONVOLVE is run, rather than using CLOSEUP before CONVOLVE.

2)After a command has been entered, it may take a significant amount of time (up to 5 minutes) before the display is complete. If the command to display the data is in error, this can result in an unreasonable amount of time lost. If it is desirable to terminate a command before completion, the user can type <Control-C> to break the execution of the command, and return control to the next higher level. Striking <Control-C> twice in a row, or striking <Control-Y> will terminate the program.

3) CONVOLVE contains special code to do single IRM convolutions. For this reason, the times for doing single and multiple convolutions are not linearly related.

4) While the DeAnza does contain hardware to do image manipulation of the types described in this routine, all computations are all done in the VAX. This is to allow extended numerical accuracy while computing the convolution sums.

5.4 Example

To do a simple 3x3 convolution operation on the image in IRMO, the following sequence would be used.

```
$CONVOLVE
convolve>G
```

.........The following image would be displayed on the screen..........

```
+---------+---------+---------+
| w0=     | w0=     | w0=     |
| w1=     | w1=     | w1=     |
| w2=     | w2=     | w2=     |
| w3=     | w3=     | w3=     |
+---------+---------+---------+
| w0=     | w0=     | w0=     |
| w1=     | w1=     | w1=     |
| w2=     | w2=     | w2=     |
| w3=     | w3=     | w3=     |
+---------+---------+---------+
| w0=     | w0=     | w0=     |
| w1=     | w1=     | w1=     |
| w2=     | w2=     | w2=     |
| w3=     | w3=     | w3=     |
+---------+---------+---------+
```

.... The arrows on the VT100 or VT52 would then be used to move the cursor around to fill in the weights for w0 to get the following display, then <CONTROL-Z> would be pushed.

```
+---------+---------+---------+
| w0=-1   | w0=-1   | w0=-1   |
| w1=     | w1=     | w1=     |
| w2=     | w2=     | w2=     |
| w3=     | w3=     | w3=     |
+---------+---------+---------+
| w0=-1   | w0=8    | w0=-1   |
| w1=     | w1=     | w1=     |
| w2=     | w2=     | w2=     |
| w3=     | w3=     | w3=     |
+---------+---------+---------+
| w0=-1   | w0=-1   | w0=-1   |
| w1=     | w1=     | w1=     |
| w2=     | w2=     | w2=     |
| w3=     | w3=     | w3=     |
+---------+---------+---------+
```

```
convolve>C
```

And the convolution would be performed. Results would, be displayed in IRM2, as that is the default, however they could have been displayed in IRM0, if desired.

Typical 3x3 operations on a single image plane require around 50 seconds to perform. For two image planes, the time goes up to 2 minutes.

6. CURSOR

CURSOR is a program designed to allow for simple interactive control of the dual cursor generator in the IP8500 image processor. With single line CURSOR commands, almost any valid cursor display can be generated and displayed. Cursor shape, color, and blinking patterns are typical of the controls available through CURSOR. The simple, flexible command syntax makes it easy to inquire about the status of either or both of the cursors.

There are two modes in which CURSOR accepts commands. In single command mode the command name CURSOR is typed followed by one or more command options. When all the options have been satisfied, DCL command is returned to the terminal. One use for this mode is in issuing short terse commands, such as clearing the screen of cursors:

```
$CURSOR OFF
```

The other mode for using CURSOR allows for multiple cursor commands to be issued efficiently. In multiple command mode, the command name is typed with no options. CURSOR then replys with a prompt. Lines containing options can then be typed at the terminal. Each line containing options can then be typed at the terminal. Each line is acted upon, followed by a return to the prompt, as in the following:

```
$CURSOR
cursor> ONE BLINK TWO NO BLINK
cursor> STATUS
.    .    .
```

6.1 CURSOR Command Options

The following are the command options for CURSOR grouped for convenience by logical functions.

Option Group	Keywords	Actions
Selection	ONE,CONE,1,C1	Sets default cursor to be acted on by later command options to cursor 1.
	PONE,P1,PCONE,PC1	Sets the default cursor to be acted on by command options to be programable cursor 1.

	TWO,CTWO,2,C2	Sets default cursor to be acted on by later command options to cursor 2.
	PTWO,P2,PCTWO,PC2	Sets the default cursor to be acted on by command options to be programable cursor 2.
Display	ON	Turns on (i.e. makes visible) the default cursor(s).
	OFF	Turns off (i.e. makes invisible) the default cursor(s).
	MODE n	Sets the cursor display mode to the value N. Display mode controls the shape of the cursors. The range of N is -18 to 7. If a question mark ("?") is entered, the current mode is shown.
	CHECKERboard n	For solid cursor modes, the cursor display is a pattern of squares n pixels on a side 0<=n<=15
	X_CHECKER n	For solid cursor modes, the cursor checking is enabled, with the horizontal checker size set to n pixels. As before, 0<=n<=15
	Y_CHECKER n	For solid cursor modes, the cursor checking is enabled, with the vertical checker size set to n pixels. As before, 0<=n<=15
Blinking	BLINK - NO BLINK	Controls whether the default cursor is to have blinking enabled or not. Blinking causes the cursor to shift from logical white to logical black at some rate.
	RATE n	Controls the frequency at which the cursors can shift from logical black to logical white. Both

		cursors flash at the same rate. 0<=n<=15
	ALTERNATE -	Controls whether the cursors both shift from logical black to logical white at the same time, as opposed to one being logical white while the other is logical black.
Spatial	POSITION nx ny	The location of the default cursor is set to the values nx and ny. Both values must be in the range 0 to 511.
Report	STATUS	Causes a display of the cursor state to be sent to the terminal.
	MODE ?	The current mode is displayed on the terminal.
	X_CHECKER ?	The current horizontal checker size is displayed.
	Y_CHECKER ?	The current vertical checker size is displayed.
	RATE ?	The current frequency of changes from logical white to logical white is displayed.
	LOCATION	The current location of the cursors is printed.
Color Control	LOGICAL_WHITE	Sets the default for logical color to be logical white.
	LOGICAL_BLACK	Sets the default for logical color to be logical black.
	DIM	Modifies colors to be half intensity.
	AQUAmarine	Selects the color to be substituted for the default logical color to be aqua.
	BLACK	Selects the color to be substituted for the default logical color to be black.
	BLUE	Selects the color to be substituted for the default logical color to be blue.

CLEAR	Selects the logical color to not be shown, I.e. the cursor is transparent. DIM is not a valid modifier for clear.
GREEN	Selects the color to be substituted for the default logical color to be green.
PURPLE	Selects the color to be substituted for the default logical color to be purple.
RED	Selects the color to be substituted for the default logical color to be red.
WHITE	Selects the color to be substituted for the default logical color to be white.
YELLOW	Selects the color to be substituted for the default logical color to be yellow.

6.2 Notes on Combining Options

0. All options are separated by blanks, slashes, or plus signs.

2. Cursor selection. By default, if no selection of cursor has been made, both cursors will be affected. For example the command:

```
$CURSOR OFF
```

will turn off both cursors, while the command:

```
$CURSOR ONE OFF
```

will turn off only cursor one.

3. Color selection. Color and logical intensity selection man be made in either order. The following two commands have the same effect:

```
$CURSOR LOGICAL_BLACK RED
$CURSOR RED LOGICAL_BLACK
```

5. Color selection. If the word DIM appears before a color, the color is shown at a reduced intensity. The only exceptions to this are that DIM CLEAR is the same as CLEAR, and DIM BLACK is actually a slate.

6. Color selection. If the color selected for logical black and logical white is the same, you will never be able to observe blinking.

7. General. Options may be set which have no effect on the current display. For example, the alternate option may be in effect, but will not alter the display unless there is blinking.

8. General. Comments can be included on any line after an exclamation mark ("!"). This is sometimes useful in documenting command files.

7. HISTO

HISTO is a program to do various forms of histogram equalization,and image intensity quantization studies. With HISTO it is possible to do image histogram modifications with a minimum of time and keystrokes.

When HISTO is started, cursor number 1 is set to the full screen flashing cross. Since the default image refresh memory to modify is IRMO (which is usually shown in red) the cursor flashes from white to red. If the image refresh memory under study is changed (via the "0","1","2", or "3" commands) the color of the flashing cursor changes as a visual reminder of the IRM under study. For IRMO, the cursor flashes white to red, IRM1, white to green, IRM2, white to blue, and IRM3, white to black. The joystick can be used to move the cursor around on the screen.

Some commands ("R" and "C") cause an intensity versus position plot to be drawn. In this case, the row and column displayed are the row or column indicated by the cursor position.

When the window ("W" command) is used to enter values, the cursor shape is changed to the flashing rectangle. The cursor box can be moved around by using the joystick with both cursor 1 and cursor 2 switches in the ON position. The shape of the cursor box can be changed by turning off one of the cursor switches and moving one corner of the box with respect to the other.

Some commands give a plot of a histogram on the DeAnza monitor screen in the graphics channel (IRM3). If no graphics output is desired the "*" command can be used to toggle off or on the plot option. By default, when IRMO is under study the graphics are drawn in blue-green, when IRM1 is under study the graphics are drawn in purple, and when IRM2 is under study the graphics are drawn in yellow.

7.1 Commands

Commands are of the form:

A - Draw in all colors (White)
for histogram plots or row/column intensity cross sections
B - Draw histogram in Blue lines
for histogram plots or row/column intensity

cross sections
C - Column histogram chart of position vs. intensity for the column shown by the cursor. This column is used as the sample for the transformations done by the T command.
D - Define a new color to draw with for histogram plots or row/column intensity cross sections
E - End program, also the space bar will terminate the program
F - Fix a transformation. Not implemented yet on the IP8500.
G - Draw histogram in Green lines for histogram plots or row/column intensity cross sections
H - Tell the user about the program commands interactively
I - Intensity histogram of row designated by the cursor
J - Intensity histogram of column designated by the cursor
K - Kill (clear) graphics overlay, useful when the output display becomes cluttered
Q - Quantization to limited steps. For posterization studies. Explained in detail elsewhere.
R - Row histogram chart of position vs. intensity for the row shown by the cursor. This column is used as the sample for the transformations done by the T command.
T - Perform one of various transforms. More on this later.
W - Use cursor window to enter statistics for both intensity histogram and transform.
Z - Draw histogram in Red lines for histogram plots or row/column intensity cross sections
0 - For sampling, histogram uses IRM0 (red)
1 - For sampling, histogram uses IRM1 (green)
2 - For sampling, histogram uses IRM2 (blue)
3 - For sampling, histogram uses IRM3 (graphics overlay)
$ - Execute CLI command
@ - Execute histo input command file
! - Comment line
. - Toggle on and off terminal display
* - Toggle on and off graphics display

7.2 Transformations

Intensity transformations are useful when the entire dynamic range of display intensities is not used to best advantage. For example,when an area of a picture has too low of a contrast range, it may be possible to apply one or more intensity transformations to the region of interest to bring out hidden detail. No "new"information is added to a picture by transforming the intensities, however it is possible to display the information in a much more meaningful manner with transformations.

The mathematical basis for the various transformations produced by this program can be found in William Pratts book "Digital Image Processing"

Valid Transformations are:

```
A ......Absolute (Data displayed as stored)
En......Exponential (typically, n is around .01)
H ......Provide on-line help
L ......Hyperbolic logarithmetic
N ......Negative (Reverse of current)
Rn......Rayleigh (typically, n is around 20)
S ......Show current as a plot in the graphics
U ......Uniform
+n......Add constant to current, where n is the
       constant
-n......Subtract constant from current, where n is
        the constant
*n......Multiply current by constant, where n is the
        constant
/n......Divide current by constant, where n is the
        constant
>n......Threshold current below, where n is the
        constant
<n......Threshold current above, where n is the
        constant
```

Normally, operations are done the moment the keystroke is complete, however on those operations requiring numeric input, a carriage return is necessary to terminate the numeric input.

Transformations are of the form:

```
histo>TR20.<return>
```

7.3 Quantizations

Intensity quantization studies are useful in studies where it is useful to know how many bits of significance the data has, in a simple and not very rigorous manner. Many times it is useful to look at data with reduced quantization levels to observe the human factors that are at work in image analysis.

Valid Quantizations are as follows:

```
Qn .....Quantize current IRM to n levels
QC .....Quantize to levels set continuously by cursor
QH .....Get on-line help
Q? .....Report current quantization levels.
```

When continuous quantization is desired (histo>QC), the cursor input is terminated by holding down the <control>key and pressing the letter 'C'.

7.4 Command Files

HISTO does allow for command files. In command files, all cursor input is replaced by numeric input on the line of the command. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the HISTO commands.

For example, to get a position vs. intensity chart for row 300 in a command file, you would enter the command:

```
R300
```

7.5 Notes

1) The cursor positioning in HISTO assumes that the picture is neither scrolled nor zoomed. For this reason, it is necessary to run CLOSEUP on pictures after HISTO is run, rather than using CLOSEUP before HISTO.

2) HISTO uses the intensity look-up tables in the DeAnza to transform the pictures in a minimum of time. For this reason, the transformations do not affect the values of data stored.

8. ISOINT

ISOINT is a program that can produce pictures which have all or a portion of the gray scale mapped into a pseudocolor display where the intensity of the range in question is mapped into a range of colors, each color being of equal intensity.

In order for this program to run correctly, the same image must be loaded into three channels, 0, 1, and 2. The MATH program can be used to do this.

The horizontal position of the vertical cursors determines the range of intensities to be mapped into colors, while the height of the cursors determines the value of the iso-intensity to be displayed. If the cursor is to the extreme left, intensity 0 is mapped into solid blue. If a cursor is to the extreme right, intensity 255 is mapped into solid red. The intensity half way between the cursors is mapped into solid green.

One way to think of the cursor control of color is as follows. Imagine the horizontal base of the display as an intensity axis ranging from 0 to 255, and the vertical side being an axis a second intensity range also ranging from 0 to 255 (the (0,0) point is the lower left hand corner of the screen, and the (255,255) point is the upper right hand side of the screen). The vertical bars of the two cursors define a domain of values on the horizontal base over which the color mapping is done. Moving the cursors closer together shortens the range, while moving both cursors at the same time changes the domain over which the color mapping is done. The single horizontal bar defines the iso-intensity into which the domain is mapped. Moving the horizontal bar to the top of the screen causes the output to be at intensity 255, while at the bottom the output would be intensity 0.

8.1 Commands

Valid commands are as follows:

```
E .....End the program
H .....On line help
L .....Linear interpolation
S .....Sineusoid interpolation
```

8.2 Example

Assuming there was a mono-color picture named PIC.B1, which had a range of interesting information from intensity 256 to 384. To display this information as a function of color, the following steps would be executed. First, the MATH program would first be used to place the image into IRM0, IRM1, and IRM2. MATH would then be exited, and the ISOINT program would be entered. While ISOINT is running, the cursors would be moved such that one cursor was half way across the screen horizontally, while the second cursor was moved to the three quarters horizontal mark. The display would show the image in full pseudo-color.

The commands to do the above are as follows:

```
$MATH
math>0=pic.b1
math>1=pic.b1
math>2=pic.b1
math>end
$ISOINT

<move the cursors to the desired location>

.
.
.
```

9. JOYSTICK

JOYSTICK is the first attempt at an interactive computer graphics system for the DeAnza. It allows for straight lines and curves to be draw on in the graphics overlay. Lines can be of any color. Single point deletion is allowed.

9.1 Commands

B	......Draw in blue
D	......Draw line to this point from previous point
E	......End the program
G	......Draw in green
H	......On line help
M	......Move the light to this point without drawing a line
R	......Draw in red
T	......Trace the joystick motion to give a curve
X	......Delete last point drawn

9.2 Notes

As this program is a stop-gap program and scheduled for replacement at a near future date, it does not allow for all of the functionality that should be included.

10. LABEL

Label is an interactive program to support the hardware character generator on the IP8500. This program assumes the existence of the alphanumerics generator option. The hardware character generator allows for the monitor to display 25 lines of 80 characters each overlayed on the image data.

With LABEL it is possible to annotate images or graphics as produced by the image processor with a minimum of difficulties. The screen design option (S) provides a one to one mapping from the display of the VT100 or VT52 terminal to the display on the image processor. The arrow keys can be used to move the cursor on the display face, and the keyboard used interactively to enter data.

10.1 Commands

Commands are of the form:

```
C.......Clear the screen of the display
E.......End the program
H.......Interactive Help
J.......Use the Joystick to point to a place on the
        screen for starting display of labels
Mn......Mode of text display
        n=0 ..... White letters on Black
          1 ..... Yellow letters on Black
          2 ..... Black letters on White
          3 ..... Black letters on Yellow
S.......Screen Design
$.......Execute DCL command
@.......Do label command file
!.......Comment line.
. ......Toggle on/off terminal output
```

Normally, operations are done the moment the keystroke is complete, however on those operations requiring either numeric input or filenames, a carriage return is necessary to terminate the input.

10.2 Screen Design

When the screen design mode is entered, the terminal screen is cleared, any text now in the alphanumerics display is shown on the terminal. The arrow keys on the terminal can then be used to position the cursor to one of the 25 rows of 80 columns where the text can be typed. Data will be shown on the monitor face at the corresponding location.

In screen design mode, typing a space will cause a dark marker to appear on the display. Using an arrow to move the cursor will not change the text. The backspace key, as opposed to the delete key, should be used to remove characters from the display, or the C command can be used to clear the entire screen buffer.

To get out of screen design mode and return to LABEL command mode, type <CONTROL-Z>.

10.3 Command Files

LABEL does allow for command files. In command files, no cursor input is allowed. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the LABEL commands.

10.4 Types of operation

This program allows for two independent types of operation: single label, and interactive mode. In single label mode, any text typed after the command to evoke LABEL is displayed centered on the last line of the display of the screen. In interactive mode, any of the commands described above can be used to customize a display.

10.5 Notes

1) The display has fixed positions for the characters, so that not all positions can be used for display.

2) The alphanumerics generator has only uppercase letters, ten numbers, and a very few special symbols. Label will convert all input text into upper case automaticly, however not all special symbols can be shown. If a symbol cannot be shown, the same symbol as the space will be shown on the display monitor.

3) The program "SYSINT" as supplied by DeAnza leaves the alphanumerics display in a cluttered mess with the screen filled with meaningless symbols. After running "SYSINT", the clear command of label can be used to remove these spurious alphanumerics.

11. LANDSAT

Landsat displays a windowed (512x512 byte window) of Landsat-D data images on the DeAnza image processor. Images can be in up to 3 bands.

11.1 Commands

Commands are of the form:

```
 0.......Output single band to go to channel 0
         (RED-Default)
 1.......Output single band to go to channel 1
         (GREEN)
 2.......Output single band to go to channel 2 (BLUE)
 Cn......Enter Column number 1-6656
 E.......End program, also hitting space bar
 H.......On-Line help
 O.......Output single band image starting at
                 selected row and column
 Rn......Enter Row number 1-5457
 Sc......Scan Row (r) or Column(c)
 Xc......Enter name of scan file for channel 0
 Yc......Enter name of scan file for channel 1
 Zc......Enter name of scan file for channel 2
 $.......Execute DCL command
 @.......Do landsat command file
 !.......Comment line.
 . ......Toggle on/off terminal output
```

Normally, operations are done the moment the keystroke is complete, however on those operations requiring either numeric input or filenames, a carriage return is necessary to terminate the input.

11.2 Command Files

LANDSAT does allow for command files. In command files, all cursor input is replaced by numeric input on the line of the command. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the LANDSAT commands.

11.3 Types of operation

This program allows for two independent types of operation:single output, and scan mode. In single output mode, the input is specified with the F command. Single output mode allows for the user to specify row and column information and then look through the image in a hunt-and-peck mode. This can be useful when attempting to register two pictures in separate frames.

Scan mode uses the X, Y, and Z commands to define the input files. By then entering the commands SR or SC the images will be placed on the screen (at about one per 75 seconds) with the position being updated by one frame width per picture. Scan mode is useful when looking through an image for objects of interest when their exact location is not known.

11.4 Notes

1) If no file is specified by the X, Y, or Z commands then no output is done to those channels. This means that the picture will remain the same in that color. Since the handling of each band takes the same amount of time, it will take three times as long to show three bands as one. Scan times can be significantly reduced by scanning in only one or two bands, as opposed to all three.

2) To turn off output operations from one file to the screen, just specify the work "OFF" as the file name. In other words, to turn off the output to the red channel, enter the command

```
landsat>XOFF
```

To turn off the output to the green channel, enter the command

```
landsat>YOFF
```

To turn off the output to the blue channel, enter the command

```
landsat>ZOFF
```

As with all commands, the commands may be in upper or lower case.

12. SCAN

Scan is used to survey the data displayed on the monitor. When running, scan uses the alphanumerics in conjunction with the cursor to display the intensity. The program sets the cursor to be a flashing dot, along the left hand margin of the screen the x/y position of the cursor of the screen, and the stored intensity in the memory registers 0 to 3 is displayed.

12.1 Commands

Scan accepts the following commands from the keyboard:

```
? ......Displays the current data at the terminal
N ......Displays data about the cursor neighborhood
E ......End SCAN, Scan can also be exited by Space
        bar
```

12.2 Notes:

SCAN shows the data values at the point in question, but not always the displayed intensities. The difference is shown by using HISTO, ISOINT or other programs. These programs modify intensity lookup tables in the DeAnza hardware which affect the way an image is displayed, but not the data that is stored. In other words, what you see is not always what you get.

13. SURFACE

Surface produces 3-dimensional perspective contour graphs of images already displayed on the DeAnza image processor. Both the original image, and the output surface are output to the DeAnza. The graphics produced use the overlay graphics planes and do not affect the data being displayed.

13.1 Commands

Commands are of the form:

```
0.......Sample IRM 0 for plot(default)
1.......Sample IRM 1 for plot
2.......Sample IRM 2 for plot
3.......Sample IRM 3 for plot
A.......Automatically set display bounds to
        data(default)
B.......Draw plot in blue
G.......Draw plot in green(default)
E.......End program
H.......Help
K.......Kill (clear) graphics  on screen
Ln......Set lower bound on intensities displayed
Ox......Output is to be drawn to region
R.......Draw plot in red
S.......Create surface plot
Un......Set upper bound on intensities displayed
Vn......Look at graph in one of 9 viewing angles.
W.......Use cursors to get windowed area of picture
$.......Execute DCL command
@.......Do SURFACE command file
!.......Comment line
. ......Toggle on/off terminal output
```

Normally, operations are done the moment the keystroke is complete, however on those operations requiring numeric input, a carriage return is necessary to terminate the numeric input.

13.2 Output Commands

The output of the SURFACE program can go to all or part of the display monitor. The selection of the space in which the display is to go is done via the Output command.

Valid Output commands are as follows:

```
OA........Output to entire screen
OH........Produce this listing
OC........Output to region bounded by cursor
```

13.3 Command Files

SURFACE does allow for command files. In command files, all cursor input is replaced by numeric input on the line of the command. To execute a command file, enter a commercial at sign ("@") followed by the filename and type of the VAX file containing the BITS commands.

13.4 Note

1)The cursor positioning in SURFACE assumes that the picture is neither scrolled nor zoomed. For this reason, it is necessary to run CLOSEUP on pictures after SURFACE is run, rather than using CLOSEUP before SURFACE.

13.5 Example

Normally, to produce a surface plot of a portion of a picture the person evokes this plotting program, uses the W command to select a portion of the picture to analyze, sets the program to draw the *surface* in the over the entire screen (due to the low-resolution of the DeAnza), and issues the command to do the plot (the C command).The terminal session to do this would appear as follows:

```
$SURFACE
surface>W
Move the cursor box to the position
requested, and press <return> when ready

 the cursor is positioned for size and
 location here
surface>S
```

14. Demonstration Programs

Two programs have been written to show off some of the more colorful, if not useless, capabilities of the DeAnza display. This documents these programs.

These programs use the graphics overlay channel to produce pretty pictures on the monitor. Image refresh memory 3 is used as the graphics memory. To clear the graphics memory after a program has been run, use the math program in the following way:

```
$MATH 3=0<return>
```

14.1 1A

Produces a series of pictures of screen in the style of several of the modern cubistic painters. The patterns are partly random. The colors are chosen from a palet of 255 possible colors, with the possibility of letting the background picture show through as one of the possible colors. The pictures are displayed for five seconds, and then the computer continues on with the next creation.

To evoke the program, type the following in response to the system prompt.

```
$ RUN [IP]1A
```

To terminate the program, hold down the control key (lower left hand of the keyboard) and press the letter "Y" at the same time.

14.2 Spin

Produces an animated series of boxes flashing around the screen, or into/out of the center of the display. The direction of the display changes from time to time on a random basis.

To evoke the program, type the following in response to the system prompt.

```
$ RUN [IP]SPIN
```

To terminate the program, hold down the control key(lower left hand of the keyboard) and press the letter "Y" at the same time.

IP8

De Anza IP8500 Manipulation Routine

IP8500 Image Processing Subrotines

By

William Brent Lander
JAYCOR

1. Introduction

This document discusses the general De Anza handling routines that were developed at NRL code 6520 to use the Ip8500 Imagearray processor. These programs are all written in VAX Fortran Four Plus and have been shown to run on version 3.4 of VMS as supplied by Digital Equipment Corporation. These routines use the device driver (level 0) of the De Anza LIPS system, but are not related above that level.

All software described here was written by William Brent Lander.

The software described here is divided into several sections.

The section on software linkage routines describes facilities to attach the De Anza to the user's program and free the De Anza as necessary.

The section on the Alphanumerics option tells how to display data that is in text form (letters and numbers) on the output monitor.

The section on the cursor describes how to manipulate the dual cursor option. At this time, no routines are available for the programable cursors.

The section on the Intensity Transformation Tables describes how to modify the displayed image without changing the data.

The section on Memory Graphics describes how to draw lines on the individual memory channels along with the data. This is useful in making permanent changes to the data.

The section on Overlay Graphics describes a set of routines analogous to the Memory Graphics which will draw lines and curves in a graphics channel that can be displayed in color without altering the data.

The next two sections of this document describe methods for reading a picture from the De Anza image refresh memories, and writing pictures to the image refresh memories.

The final section of this document describes the video output controller. The video output controller allows for pseudo coloring on picture segments, and the mapping between image refresh memories and the monitor display.

2. Software Linkage Routines

The routines in this segment describe the high level linkage between the operating system and the image processing software that is necessary to the user. In general these routines will not be needed directly, but are included in the appropriate subroutines.

2.1 Subroutine Ip8Chan(Chan)

Purpose:

This subroutine causes a software link to be established to the De Anza. It is through this link that the actual commands to the registers of the Imagearray Processor are loaded.

Parameter:

Chan - Integer*2 - Returns the number of the channel that was assigned to the device.

Program Author: William Brent Lander, JAYCOR.

Date: March 1, 1983

2.2 Subroutine Ip8Free

Purpose:

This subroutine releases the software link established to the De Anza. Freeing this link allows other programs or users to gain access to the De Anza.

Parameters:

None.

Program Author: William Brent Lander, JAYCOR.

Date: March 1, 1983

3. Alphanumerics Options Routines

The subroutines described in this section can be used to annotate the display sent to the monitor.

3.1 Subroutine Ip8A_Write(Irow, Icol, Text, Mode)

Purpose:

To allow for the use of the alphanumerics overlay option of the IP8500 by a Fortran program. This program transfers data to the alphanumerics display option.

Parameters:

```
        Irow - Integer*4 - row number for text transfer
                 to begin on.
        Icol - Integer*4 - column number for text
                 transfer to begin on.
        Text - Character*(*) - text to be displayed.
        Mode - Integer*4 - the data display mode for
                 the overlay
                          mode=0--white on black.
                              =1--yellow on black.
                              =2--black on white.
                              =3--black on yellow.
                              =4--white on image.
                              =5--yellow on image.
                              =6--image on white.
                              =7--image on yellow.
                              =8--black on image.
                              =9--black on image(same as 8).
                              =10-image on black.
                              =11-image on black(same as 10).
                              =12-black on white.
                              =13-black on white.
                              =14-white on black.
                              =15-white on black.
```

Program Author: William Brent Lander, Jaycor.

Date: March 29, 1983

Notes:

1) The alphanumerics option has built in rap-around for text extending past the last column in a row. If text attempts to extend past the end of row, the text will be continued on the following column.

2) This subroutine will not allow for transfers to be outside the 2000 character alphanumeric memory window.

3) Writing nulls (ascii character 0) will allow the screen to be visible through the text.

4) The first row (row 1) is at the top of the screen. The first column (column 1) is at the left hand side of the screen.

5) This routine has a call to IP8V_Alphamode to turn of the graphics and alphanumerics display. Due to hardware constraints, both of these (graphics and alphanumerics) must be turned on at the same time.

3.2 Subroutine Ip8A_Off

Purpose:

This routine turns off the alphanumeric overlay on the IP8500 without destroying the current contents of the alphanumerics display. In this way, the display can be removed and replaced in a single call.

Parameters:

None.

Program Author: William Brent Lander, Jaycor

Date: September 10, 1983

Note:

1) This routine only clears the bit in the Control and status register so that no display is made. The contents of the memory remain the same.

2) The routine IP8A_WRITE can be used to load the display memory with the alphanumerics turned off, and then IP8A_ON can be used to flash the completed display to the monitor. In this way the programmer can produce the effect of instantaneous display update.

3)To Clear the memory, use the routine IP8A_CLEAR.

3.3 Subroutine Ip8A_On

Purpose:

This subroutine turns on the alphanumeric overlay on the Ip8500. The contents of the alphanumeric memory are not affected by this toggling, only the display to the monitor.

Parameters:

None.

Program Author: William Brent Lander, Jaycor

Date: March 31, 1983

Note:

1)This routine only sets the bits in the VOC register so that display is shown. The contents of the memory remain the same. This VOC register modification is done by the IP8V_AlphaDisabl routine.

2)To Clear the memory, use the routine IP8A_CLEAR.

3.4 Subroutine Ip8A_Clear

Purpose:

This subroutine is used to set the contents of the alphanumeric overlay completely to nulls. This results in the removal of all text from the screen, without changing the Control and Status Register. The contents of the alphanumeric register is set to ASCII character 0 (Null).

Parameters:

None.

Program Author: William Brent Lander, Jaycor.

Date: March, 1983.

3.5 Subroutine Ip8A_Read(Irow, Icol, Text, Length)

Purpose:

This subroutine is used to allow for the reading of the alphanumeric overlay program by a fortran program.

Parameters:

Irow - Integer*4 - Row number for text transfer to begin on.
Icol - Integer*4 - Column number for text transfer to begin on.
Text - Character*(*) - Text to be displayed.
Length - Integer*4 - Number of characters to transfer

Program Author: William Brent Lander, Jaycor.

Date: March 29, 1983

Notes:

1) The device has built in rap-around when reading text extending past the last column in a row. If reads are allowed to extend past the end of row, the returned will be the text continued on the following column.

2) This subroutine will not allow for transfers to be outside the 2000 character alphanumeric memory window.

3) Reading nulls (ascii character 0) will correspond to regions where the contents of the image refresh memories is visible through the text.

4. Cursor Input Output Routines

These routines are useful in getting service from the cursor option. With the cursor option, a user can point to a segment of the screen to indicate regions of interest.

4.1 Subroutine Ip8C_Read(Ix1, Iy1, Ix2, Iy2)

Purpose:

This subroutine allows a Fortran program to easily obtain the current cursor position from the screen. The cursor positions themselves can be controlled either by program control (see the IP8C_WRITE routine) or by external device, such as the Joystick, or Trackball.

Parameters:

Ix1 - Integer*4 - Horizontal location of cursor number 1.
Iy1 - Integer*4 - Vertical location of cursor number 1.
Ix2 - Integer*4 - Horizontal location of cursor number 2.
Iy2 - Integer*4 - Vertical location of cursor number 2.

Note:

All cursor positions are to be in the range 0 to 511, where the (0, 0) point is the lower left hand side of the screen.

Program Author: William Brent Lander, Jaycor.

Date: March 29, 1983

4.2 Subroutine Ip8C_Write(Ix1, Iy1, Ix2, Iy2)

Purpose:

With this subroutine, a program can control the absolute locations of the two cursors. This is useful in setting the size of a box (when using two cursors) or moving the cursors to indicate some region of interest that the program has found.

Parameters:

Ix1 - Integer*4 - Horizontal location of cursor number 1.
Iy1 - Integer*4 - Vertical location of cursor number 1.
Ix2 - Integer*4 - Horizontal location of cursor number 2.
Iy2 - Integer*4 - Vertical location of cursor number 2.

Note:

All cursor positions are to be in the range 0 to 511, where the (0, 0) point is the lower left hand side of the screen.

Program Author: William Brent Lander, Jaycor.

Date: March 29, 1983

4.3 Subroutine Ip8C_Control (Mode, Cursor1, Cursor2, Blink1, Blink2, Fast)

Purpose:

Using this routine it is possible to control the cursor display.

Parameters:

Mode - Integer*4 - The type of cursor display desired.0<mode<8 gives the standard cursors, mode=-1 will retain the current cursor mode. Mode=-2 to -18 will give the cursors the description in the notes.
Cursor1 - Integer*4 - If Cursor1=1 the standard cursor will be displayed, if Cursor1=-1 the programmable cursor 1 will be displayed, otherwise the cursor will not be displayed.
Cursor2 - Integer*4 - If Cursor2=1 the standard cursor will be displayed, if Cursor2=-1 the programmable cursor will be displayed, otherwise the cursor will not be displayed.
Blink1 - Integer*4 - If positive, cursor 1 willblink. If zero, no blinking will occur. If -1, blinking will remain unchanged.

Blink2 - Integer*4 - If positive, cursor 2 will blink. If zero, no blinking will occur. If -1, blinking will remain unchanged.

Fast - Integer*4 - Rate of flashing. -1<Fast<15. 0 is rapid flashing, 15 is slow flashing. If Flash=-1, no change to the current flashing rate.

Program Author: William Brent Lander, Jaycor.

Date: March 29, 1983

4.4 Subroutine Ip8C_Off

Purpose:

This subroutine resets the cursor control hardware so that no cursors will be displayed. The options for which cursor will be shown, mode of display, and blinking are not changed.

Parameters:

None.

Program Author: William Brent Lander, Jaycor.

Date: March 29, 1983

Note:

This routine calls IP8V_CursorDisa to remove the cursor.

4.5 Subroutine Ip8C_Status (Mode, Cursor1, Cursor2, Blink1, Blink2, Fast)

Purpose:

This routine complements the IP8C_CONTROL to allow for monitoring of the cursor status.

Parameters:

Mode - Integer*4 - Returns the type of cursor display 0<mode<8. If the mode is non-standard, the mode value will be in the range -2 to -18.

Cursor1 - Integer*4 - If Cursor1=1 the standard cursor is being displayed, if Cursor1=-1 the programmable cursor 1 is being displayed, otherwise the cursor is not being displayed.
Cursor2 - Integer*4 - If Cursor2=1 the standard cursor is being displayed, if Cursor2=-1 the programmable cursor is being displayed, otherwise the cursor is not being displayed.
Blink1 - Integer*4 - If positive, cursor 1 isblinking.
Blink2 - Integer*4 - If positive, cursor 2 is blinking.
Fast - Integer*4 - Rate of flashing. $0<Fast<15$.

Program Author: William Brent Lander, Jaycor.

Date: March 29, 1983

4.6 Subroutine Ip8C_PCLOAD(PCNUMBER, VECTOR)

Purpose:

The DeAnza IP8500 has two programmable cursors per cursor board. These cursors are numbered 1 and 2. The cursors consist of a 64x64 array of pixels. Each pixel may be clear(value 0), logical white(value 1), or logical black(value 2). The cursors are controled by the joystick, track ball, or computer the same as the standard cursors. The visual displayed on the screen is the primary difference between the two cursor forms.

This routine is used to load the contents of the arrays displayed as programmable cursor 1(PC1) and programmable cursor 2(PC2). The format of the information stored in the array VECTOR is discussed in the DeAnza IP8500 reference guide. The routine IP8C_MAKEPC can also be used to generate these arrays.

Parameters:

PCNUMBER - Integer - Number in the range 1 to 2 inclusive which contains the number of the programmable cursor to load.
VECTOR - Integer*4 Array of size 1024 - Array containing the information to be transfered to the DeAnza. The information is stored 4 pixels per number.

Note:

This routine loads the display registers with the requested form. It does not enable the cursor display. The cursors can be displayed using the IP8C_CONTROL routine.

4.7 Subroutine Ip8C_PCRead(PCNUMBER, VECTOR)

Purpose:

This routine is used to obtain the contents of the arrays displayed as programmable cursor 1 or programmable cursor 2. The format of the information stored in the array VECTOR is discussed in the DeAnza IP8500 reference guide.

Parameters:

PCNUMBER - Integer - Number in the range 1 to 2 inclusive which contains the number of the programmable cursor to read.
VECTOR - Integer*4 Array of size 2048 - Array containing the information to be transfered from the DeAnza. The information is stored 4 pixels per number.

Note:

This routine obtains the contents of the display registers. It does not alter the cursor display. The cursors can be displayed using the IP8C_CONTROL routine.

4.8 Subroutine Ip8C_PCmake(ARRAY, VECTOR)

Purpose:

This routine is used to produce the vectors to be loaded into the arrays displayed as programmable cursor 1 or programmable cursor 2. The format of the information stored in the array VECTOR is discussed in the DeAnza IP8500 reference guide.

The format of the information input to this routine is a 64x64 array of integer*4 variables, numbered from 1 to 64. Each element in the array has one pixel stored in it. The pixel value in the array is in the range 0 to 3, inclusive; 0 is clear, 1 is logical white, two is logical black, and 3 is clear. The position of the point under the position (32,32) is returned as the position of the cursor by IP8C_READ.

Parameters:

ARRAY - Integer*4 Array of size 64x64 - Each number in the array is in the range 0 to 3, the number of the cursor pixel to load read.

VECTOR - Integer*4 Array of size 1024 - Array containing the information to be transfered from the DeAnza. The information is stored 4 pixels per number.

Note:

This routine produces the vectors to be used in the display registers. It does not alter the cursor display. The cursors can be displayed using the IP8C_CONTROL routine, and loaded with the IP8C_PCLOAD routine.

4.9 Subroutine Ip8C_Checker(Xchecker, Ychecker)

Purpose:

This routine enables the checkering option on the solid cursor forms of the DeAnza cursors. Checkering modifies the solid areas of the cursors to be an array of alternating black and white rectangles, much like a checkerboard. The horizontal and vertical sizes of the checks is controled by the parameters.

The checkers are fixed on the screen, and moving the cursor changes the pattern below it. In other words, it as though there was a fixed checkerboard pattern superimposed on the screen, and the cursor is a window which enables it to be seen.

Parameters:

XCHECKER - Integer - Number between -1 and 15 which contains the number of pixels along a single horizontal check. If XCHECKER = -1 the previous value of the parameter remains unchanged

YCHECKER - Integer - Number between -1 and 15 which contains the number of pixels along a single vertical check. If YCHECKER = -1 then the previous value of the parameter remains unchanged.

Program Author: William Brent Lander, JAYCOR.

Date: February 14, 1984

Notes:

1)The checkers are only visible with the solid cursor forms. If you are using IP8C_CONTROL, that means modes between -10 and -18.

2)While in principle you can set up small checkers, there are some problems with flickering when the values of XCHECKER and YCHECKER get too small. This is due to the interleaved refreshing of the monitor.

3)One possible application of this feature is the addition of calibration markers on the screen. In photographs, the cursors could be set to a given number of pixels across, and the checking enabled to give scale to the picture.

4.10 Subroutine Ip8C_RCcheckerXchecker, Ychecker)

Purpose:

This routine returns values on the checkering option of the solid cursor forms of the DeAnza cursors. This routine enables a program to obtain the size of the cursor checks.

For more information on the properties of checkering, see IP8C_CHECK

Parameters:

XCHECKER - Integer - Returns a number between 0 and 15 which is the number of pixels along a single horizontal check. If XCHECKER = 0 the checking is disabled.

YCHECKER - Integer - Returns a number between 0 and 15 which contains the number of pixels along a single vertical check. If YCHECKER = 0 then the checking is disabled.

Program Author: William Brent Lander, JAYCOR.

Date: February 14, 1984

Notes:

1)The checkers are only visible with the solid cursor forms. If no solid cursor is enabled, this routine will still return the value that would be used if solid form cursor were enabled.

4.11 Subroutine Ip8C_Blink(C1Mode, C2Mode)

Purpose:

Blinking mode controls whether or not the cursors blink on even or odd pulses of the cursor rate clock. This routine allows the blinking mode of the two cursors to be independently controled.

Parameters:

C1MODE - Integer - The blinking mode control for cursor 1 and programmable cursor 1. If C1MODE=-1, the current value remains unchanged. If C1MODE=0 the cursor blinks from white to black. If C1MODE=1 the cursor blinks from black to white.

C2MODE - Integer - The blinking mode control for cursor 2 and programmable cursor 2. If C2MODE=-1, the current value remains unchanged. If C2MODE=0 the cursor blinks from white to black. If C2MODE=1 the cursor blinks from black to white.

Program Author: William Brent Lander, JAYCOR.

Date: February 14, 1984

Notes:

For example, if the parameters are set such that C1MODE=1 and C2MODE=0, the cursors will blink out of phase with each other.

4.12 Subroutine Ip8C_RBLINK(C1Mode, C2Mode)

Purpose:

Blinking mode controls whether or not the cursors blink on even or odd pulses of the cursor rate clock. This routine allows the blinking mode of the two cursors to be interrogated.

Parameters:

C1MODE - Integer - The blinking mode control for cursor 1 and programmable cursor 1. If C1MODE=0 the cursor blinks from white to black. If C1MODE=1 the cursor blinks from black to white.

C2MODE - Integer - Returns the blinking mode control for cursor 2 and programmable cursor 2. If C2MODE=0 the cursor blinks from white to black. If C2MODE=1 the cursor blinks from black to white.

Program Author: William Brent Lander, JAYCOR.

Date: February 14, 1984

Notes:

For example, if the parameters are set such that C1MODE=1 and C2MODE=0, the cursors are blinking out of phase with each other.

5. The Image Plane Memory Graphics

The image plane memory graphics routine write lines and boxes directly into the data planes in which data is stored. This is useful in making changes to the data itself, and not just overlay graphics.

It is useful to contrast these graphics with those defined in the IP80 routines.

5.1 Subroutine Ip8G_Move(X, Y)

Purpose:

This routine moves the graphics pointer to (x,y).

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the current graphics pointer should go. The next IP8G_Draw call will cause a line to be drawn from this location.

Y - Real - Vertical location in the user's coordinate system of where the current graphics pointer should go. The next IP8G_Draw call will cause a line to be drawn from this location.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

5.2 Subroutine Ip8G_Origin(X, Y)

Purpose:

This Routine sets the origin for the users coordinate system to the point (x,y)

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the current graphics origin should be. Future calls to IP8G_Draw and IP8G_Move will be made relative to this location.

Y - Real - Vertical location in the user's coordinate system of where the current graphics origin should be. Future calls to IP8G_Draw and Ip8G_Move will be made relative to this location.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

5.3 Subroutine Ip8G_Scale(X, Y)

Purpose:

This routine sets scaling factor for further calls to routines using the user coordinate set, as opposed to the absolute coordinate set.

Parameters:

X - Real - Horizontal scaling factor applied to the user's coordinate system. Future calls to IP8G_Draw and Ip8G_Move will be scaled by this factor.
Y - Real - Vertical location in the user's coordinate system of where the current graphics pointer should go. Future calls to IP8G_Draw and Ip8G_Move will be scaled by this factor.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

5.4 Subroutine Ip8G_Draw(X, Y)

Purpose:

This routine causes a line, in the users coordinate system to be drawn from the current graphics pointer location to point (X, Y). The current graphics pointer location is set by a previous call to IP8G_Move, IP8G_Draw, IP8G_Amove, or IP8G_Adraw.

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the a line from the current graphics pointer should end. The next IP8G_Draw call will cause a line to be drawn from this location.

Y - Real - Vertical location in the user's coordinate system of where the a line from the current graphics pointer should end. The next IP8G_Draw call will cause a line to be drawn from this location.

Note:

1) The line is drawn in the color specified by IP8G_Color.

2) Clipping is done at the edge of the hardware screen.

3) Windowing is also done before the data is plotted.
Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

5.5 Subroutine Ip8G_Window(X1,X2,Y1,Y2, Ier)

Purpose:

This routine is used to set the clipping window. Requests to draw lines outside this window will be ignored. If a requested line passes through this window, only the portion of the line inclosed by the window is drawn.

Parameters:

X1 -Real- Left side of the window in screen coordinates.
X2 -Real- Right side of the window in screen coordinates.
Y1 -Real- Top of the window in screen coordinates
Y2 -Real- Bottom of the window in screen coordinates

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

Note:

1) Screen coordinates vary from 0 to 511. and are not always the same as the user coordinates.

5.6 Subroutine Ip8G_Adraw(X, Y)

Purpose:

This routine draws a line beginning at the current graphics pointer position to the screen coordinates (X, Y). The current graphics pointer position is set to the location (X, Y)

Parameters:

X - Integer - Horizontal location in the screen coordinate system of where the a line from the current graphics pointer should end.
Y - Integer - Vertical location in the screen coordinate system of where the current graphics pointer should go.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

Note:

Remember the screen coordinates are limited by the range 0 to 511.

5.7 Subroutine Ip8G_Amove (X, Y)

Purpose:

With this routine, the graphics cursor current position is set to the screen coordinate system location (X,Y).

Parameters:

X - Integer - Horizontal location in the screen coordinate system of where the current graphics pointer should be moved to.
Y - Integer - Vertical location in the screen coordinate system of where the current graphics pointer should go.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

5.8 Subroutine Ip8G_Ascreen(X, Y, Sx, Sy)

Purpose:

This subroutine returns the absolute screen coordinates corresponding to the user coordinates points.

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the data point is.
Y - Real - Vertical location in the user's coordinate system of where the data point is.
Sx - Integer - Vertical location in the screen coordinate system, of where the data point will be displayed.
Sy - Integer - Horizontal location in the screen coordinate system, of where that data point would be displayed.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

5.9 Subroutine Ip8G_Segment(Xa,Ya,Xb,Yb)

Purpose:

This subroutine uses the hardware write vector instruction of the De Anza to generate lines with the processor. This program does not use the graphics overlay capability of the system. To use the graphics overlay system, reference the IP8O_ series of subroutines.

Parameters:

Xa-Integer*4 - Value of starting x coordinate
Ya-Integer*4 - Value of starting y coordinate
Xb-Integer*4 - Value of stopping x coordinate
Yb-Integer*4 - Value of stopping y coordinate.

For all coordinates: 0<= coordinate <= 511

Program Author: William Brent Lander, Jaycor.

Date: March 9, 1983.

Note:

The graphics screen is assumed to be addressed as follows:

```
(0,511) +--------------------+ (511,511)
        I                    I
        I                    I
        I                    I
        I                    I
        I                    I
        I                    I
        I                    I
        I                    I
        I                    I
        I                    I
(0,0)   +--------------------+ (511,0)
```

5.10 Subroutine Ip8G_Color(Irm0, Irm1, Irm2, Irm3)

Purpose:

This subroutine allows for various colours to be used to fill the lines.

Parameters:

Irm0-Integer	Intensity to be written to plane 0
Irm1-Integer	Intensity to be written to plane 1
Irm2-Integer	Intensity to be written to plane 2
Irm3-Integer	Intensity to be written to plane 3

Note:

1) By setting the Intensity to a negative number, the planes are not affected by any Ip8G type of command.

2) Default value is 255 for all planes.

Program Author: William Brent Lander, Jaycor.

Date: April 6, 1983

5.11 Subroutine Ip8G_ABox(Ix1, Iy1, Ix2, Iy2)

Purpose:

To put up a box in the colored memory plane.

Parameters:

Ix1 - Integer - Column number of the first column in the region to be colored.
Iy1 - Integer - Row number of the first row in the region the region to be colored.
Ix2 - Integer - Column number of the last column in the the region to be colored.
Iy2 - Integer - Row number of the last row in the region the region to be colored.

Program Author: William Brent Lander, JAYCOR

Date: May 18, 1983

6. Intensity Transformation Table Routines

The purpose of these routines is to provide a method of controlling the intensity transformation tables in the De Anza Ip8500. Using these intensity transformation tables, it is possible to modify the displayed intensities of images without changing the data values themselves. This can be very useful in histogram modification work.

6.1 Subroutine Ip8I_Threshold (Min, Max, Irm0, Irm1, Irm2, Irm3, Irm4, Irm5)

Purpose:

To allow for the Intensity transformation planes to be loaded from a program with a ramp function, or portion of a ramp function.

Parameters:

Min - First input intensity to be ascribed a value
Max - Last input intensity to be ascribed a value
Irm0-Irm5 - integer switch to the ITU's for image refresh memories 0-5. If negative, there is no change to that IRM. If non-negative, the ramp is loaded into the corresponding ITU for that IRM.

Note:
With this subroutine, the value assigned in the ramp corresponds to the unmodified intensity.

6.2 Subroutine Ip8I_Load (JTable, Irm0, Irm1, Irm2, Irm3, Irm4, Irm5)

Purpose:

This subroutine makes it easy to load a selected set of image refresh memories with a user-supplied intensity transformation unit (ITU)

Parameters:

JTable - Integer array of 256 elements- The user supplied values to be sent to the ITU's

Irm0-Irm5 - Integer - If non-negative, the values specified by TABLE are loaded into the corresponding ITU for the memory. Otherwise, no change is made to that ITU.

Program Author: William Brent Lander, Jaycor.

Date: March 31, 1983.

6.3 Subroutine Ip8I_Ramp

```
       (Min, Max, Intenmin, Intenmax,Irm0, Irm1, Irm2,
       1                            Irm3,Irm4, Irm5)
```

Purpose:

With this subroutine a programmer is allowed to load the intensity transformation planes to be loaded from a program with a ramp function, or portion of a ramp function.

Parameters:

Min - First input intensity to be ascribed a value
Max - Last input intensity to be ascribed a value
Irm0-Irm5 - Integer switch to the ITU's for image refresh memories 0-5. If non-negative, the ramp is loaded into the corresponding ITU for that IRM. Otherwise the value is left unchanged.

Program Author: William Brent Lander, JAYCOR.

Date: March 11, 1983

6.4 Subroutine Ip8I_OFF

```
       (Irm0, Irm1, Irm2, Irm3)
```

Purpose:

To clear the video modify bits of the Memory Port Control This will result in the intensities being displayed being those in memory with no modification. In other words, what you see is what you get in the display.

Parameters:

Irm0 - Integer - If positive, the transformed data is selected for display, Otherwise, the transformed data will not be displayed.
Irm1 - Integer - If positive, the transformed data is selected for display, otherwise, the unmodified data will be selected.
Irm2 - Integer - If positive, the transformed data is selected for display, otherwise, the unmodified data will be selected.
Irm3 - Integer - If positive, the transformed data is selected for display, otherwise, the unmodified data will be selected.

Note:

This routine does set a hard register in the De Anza, hence the results may be carried over from one use to another.

Program Author: William Brent Lander, Jaycor.

Date: March 31, 1983.

6.5 Subroutine Ip8I_Read(Intensity,Memory, Itu)

Purpose:

To allow for the contents of the intensity transformation Table (ITU) to be recovered at any time.

Parameters:

Intensity - Integer*4 - array returning the values stored in the ITU.
Memory - Integer*4 - The number of the Image Refresh memory associated with the ITU.
Itu - Integer*4 Number of the ITU to read.

Program Author: William Brent Lander, Jaycor.

Date: March 31, 1983.

7. The Joystick Control Routines

The joystick routines are used to control and monitor the state of the IP8500 Joystick option. The joystick is physically different than the cursors.

7.1 Subroutine Ip8J_SetJoy(Iboard)

Purpose:

This subroutine sets software control over which joystick control board is to be used in future calls to IP8J routines. The De Anza IP8500 allows for up to 4 joystick control boards, numbered from 0 to 3. Typically, only one board per user is allowed, so the the default value of board zero makes most calls to this routine unnecessary:

Parameters:

Iboard - Integer -Number in the range 0 to 3 for the joystick control board.

Program Author: William Brent Lander, JAYCOR

Date: January 20, 1984

Notes:

Normally, this routine will not be required. Do not feel obliged to use it.

7.2 Subroutine Ip8J_Read(Ix, Iy)

Purpose:

This subroutine returns two values relating to the amount of displacement the joystick has from the center position. This is not the same as the cursor position. These values are used by the DeAnza hardware to control the rate of cursor movement.

Parameters:

Ix - Integer - Returns the right-to-left displacement of the joystick. The value is in the range -128 to 127

Iy - Integer - Returns the top-to-bottom displacement of the joystick. The value is in the range -128 to 127.

Program Author: William Brent Lander, JAYCOR

Date: January 20, 1984

Notes:

If the CURSOR1 and CURSOR2 switches are in the ON position, the joystick displacement will be used to control the cursor movement rate. If these switches are off, no changes are made to the cursors, and the registers can be used for any function.

The slides on the side of the silver joystick lever can be used to alter the register reading of the joystick in the neutral position.

7.3 Subroutine Ip8J_Status (Cursor1, Cursor2, Track, Enter, Mode, Interrupt)

Purpose:

This routine is used to return the status of the switches along the left-hand-side of the joystick control box. A value of 1 means the switch is ON; a value of 0 means the switch is OFF.

The INTERRUPT status returns 1 if an interrupt from the joystick ENTER button is pending.

Parameters:

Cursor1 - Integer - Returns the status of the top switch on the box. If ON, moving the joystick will cause cursor 1 to be moved.

Cursor2 - Integer -Returns the status of the second switch on the box. If ON, moving the joystick will cause cursor 2 to be moved.

Track - Integer -Returns the status of the third switch on the box. If ON, moving the joystick will cause the cursor to be moved.

Enter - Integer -Returns 1 if the Push-button switch at the base of the joystick control box is currently being depressed, otherwise, it returns 0.

Mode - Integer -Returns the status of the fourth switch on the box.
Interrupt - Integer -If the returned value is 1, an interrupt from the joystick is pending.

Program Author: William Brent Lander, JAYCOR

Date: January 20, 1984

Notes:

7.4 Subroutine Ip8J_Clear

Purpose:

This routine clears an interrupt once it has been signaled by the joystick. An interrupt must be cleared before a new interrupt can be acknowledged.

Parameters:

None.

Program Author: William Brent Lander, JAYCOR

Date: January 20, 1984

Notes:

The interrupt handling routines of the IP8 package are currently under development. This routine is part of that package.

7.5 Subroutine Ip8J_Enable

Purpose:

When called, this subroutine allows the next depression of the ENTER button on the joystick control lever to signal an interrupt to the VAX. Routines are currently being developed to handle the interrupts in an intelligent manner.

Parameters:

None.

Program Author: William Brent Lander, JAYCOR

Date: January 20, 1984

Notes:

7.6 Subroutine Ip8J_Disable

Purpose:

This subroutine will stop the next depression of the ENTER key on the joystick from interrupting the VAX. Normally, the IP8J_Enable and IP8J_Disable routines can be thought of as complementary operators; what one does, the other undoes.

Parameters:

None.

Program Author: William Brent Lander, JAYCOR

Date: January 20, 1984

Notes:
This routine does not clear pending interrupts.

8. The Overlay Graphics Routines

These routines differ from those covered earlier because they use the overlay graphics hardware of the IP8500. This can result in significantly faster graphics, as well as not compromising the raw data being displayed.

8.1 Subroutine Ip80_Move(X, Y)

Purpose:

This routine moves the graphics pointer to (x,y)

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the current graphics pointer should go. The next IP80_Draw call will cause a line to be drawn from this location.

Y - Real - Vertical location in the user's coordinate system of where the current graphics pointer should go. The next IP80_Draw call will cause a line to be drawn from this location.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

8.2 Subroutine Ip80_Origin(X, Y)

Purpose:

This Routine sets the origin for the users coordinate system to the point (x,y)

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the current graphics origin should be. Future calls to IP80_Draw and IP80_Move will be made relative to this location.

Y - Real - Vertical location in the user's coordinate system of where the current graphics origin should be. Future calls to IP80_Draw and Ip80_Move will be made relative to this location.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

8.3 Subroutine Ip80_Scale(X, Y)

Purpose:

This routine sets scaling factor for further calls to routines using the user coordinate set, as opposed to the absolute coordinate set.

Parameters:

X - Real - Horizontal scaling factor applied to the user's coordinate system. Future calls to IP80_Draw and Ip80_Move will be scaled by this factor.

Y - Real - Vertical location in the user's coordinate system of where the current graphics pointer should go. Future calls to IP80_Draw and Ip80_Move will be scaled by this factor.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

8.4 Subroutine Ip80_Draw(X, Y)

Purpose:

This routine causes a line, in the users coordinate system to be drawn from the current graphics pointer location to point (x,y). The current graphics pointer location is set by a previous call to IP80_Move, IP80_Draw, IP80_Amove, or IP80_Adraw.

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the a line from the current graphics pointer should end. The next IP80_Draw call will cause a line to be drawn from this location.

Y - Real - Vertical location in the user's coordinate system of where the a line from the current graphics pointer should end. The next IP80_Draw call will cause a line to be drawn from this location.

Note:

1) The line is drawn in the color specified by IP80_Color.

2) Clipping is done at the edge of the hardware screen.

3) Windowing is also done before the data is plotted.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

8.5 Subroutine Ip80_Window(X1,X2,Y1,Y2, Ier)

Purpose:

This routine is used to set the clipping window. Requests to draw lines outside this window will be ignored. If a requested line passes through this window, only the portion of the line inclosed by the window is drawn.

Parameters:

X1 -Real- Left side of the window in screencoordinates.
X2 -Real- Right side of the window in screen coordinates.
Y1 -Real- Top of the window in screen coordinates
Y2 -Real- Bottom of the window in screen coordinates

Note:

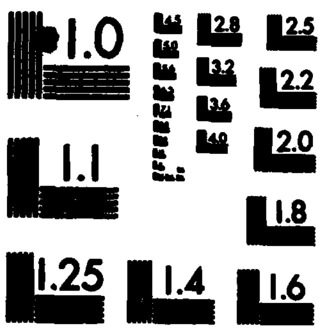
1) Screen coordinates vary from 0 to 511. and are not always the same as the user coordinates.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

AD-A146 392 INVESTIGATIVE FINDINGS OF IMAGE ENHANCEMENT TECHNIQUES 2/2
(U) JAYCOR ALEXANDRIA VA W B LANDER 26 MAR 84
JAYCOR-J206-84-004/6238 SBI-AD-E001 736
UNCLASSIFIED N00014-83-C-2273 F/G 17/8 NL

END
FILMED
DTIC



'OPY RESOLUTION_ TEST CHART

8.6 Subroutine Ip80_Adraw(X, Y)

Purpose:

This routine draws a line beginning at the current graphics pointer position to the screen coordinates (X, Y). The current graphics pointer position is set to the location (X, Y)

Parameters:

X - Integer - Horizontal location in the screen coordinate system of where the a line from the current graphics pointer should end.
Y - Integer - Vertical location in the screen coordinate system of where the current graphics pointer should go.

Note:
Remember the screen coordinates are limited by the range 0 to 511.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

8.7 Subroutine Ip80_Amove (X, Y)

Purpose:

With this routine, the graphics cursor current position is set to the screen coordinate system location (X,Y).

Parameters:

X - Integer - Horizontal location in the screen coordinate system of where the current graphics pointer should be moved to.
Y - Integer - Vertical location in the screen coordinate system of where the current graphics pointer should go.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

8.8 Subroutine Ip80_Ascreen(X, Y, Sx, Sy)

Purpose:

This subroutine returns the absolute screen coordinates corresponding to the user coordinates points.

Parameters:

X - Real - Horizontal location in the user's coordinate system of where the data point is.
Y - Real - Vertical location in the user's coordinate system of where the data point is.
Sx - Integer - Vertical location in the screen coordinate system, of where the data point will be displayed.
Sy - Integer - Horizontal location in the screen coordinate system, of where that data point would be displayed.

Program Author: William Brent Lander, JAYCOR.

Date: May 10, 1983.

8.9 Subroutine Ip80_Segment(Xa,Ya,Xb,Yb)

Purpose:

This subroutine uses the hardware write vector (WV) instruction of the IP8500 to approximate the use of true vectors with the hardware.

Parameters:

Xa-Integer*4 - Value of starting x coordinate
Ya-Integer*4 - Value of starting y coordinate
Xb-Integer*4 - Value of stopping x coordinate
Yb-Integer*4 - Value of stopping y coordinate.

For all coordinates: 0<= coordinate <= 511

Note:

The graphics screen is assumed to be addressed as follows:

```
(0,511) ------------------- (511,511)
```

```
               I                    I
               I                    I
               I                    I
               I                    I
               I                    I
               I                    I
               I                    I
               I                    I
               I                    I
               I                    I
       (0,0)   +------------------+ (511,0)
```

Program Author: William Brent Lander, Jaycor.

8.10 Subroutine Ip80_Color(Ired, Igreen, Iblue)

Purpose:

This subroutine allows for various colours to be used to fill the lines.

Parameters:

```
Ired  -Integer*4  Intensity to be written to red
Igreen-Integer*4- Intensity to be written to green
Iblue -Integer*4- Intensity to be written to blue
```

By setting the intensity to a negative number, the planes are not effected by any Ip80_ type of command.

Note:

1) Default value is 255 for all planes.

2) Due to hardware constraints, only 255 colors may be active at any one time.

3) The available colors have a resolution of only 15 intensities. This routine tries to make as good of a fit as possible to the desired color. For this reason, colors which are closely aligned may be shown as the same color.

Program Author: William Brent Lander, Jaycor.

Date: April 6, 1983

8.11 Subroutine Ip80_LoadColor(Jarray,Ncolors)

Purpose:

This subroutine allows for a user to define their own plate of colors to be used by the De Anza overlay graphics routine.

Parameters:

Jarray - Integer*4 - Array of values to be loaded into the lookup tables.
Ncolors - Integer*4 - Number of values to be loaded.

Program Author: William Brent Lander, JAYCOR.

Date: April 6, 1983

8.12 Subroutine Ip80_SaveColor(Iarray)

Purpose:

This routine will retrieve the color lookup table that is active on the current graphics plane.

Parameter:

Iarray - Integer - Array of colors returned. This array must be dimensioned 256 or larger

Program Author: William Brent Lander, JAYCOR.

Date: April 6, 1983

8.13 Subroutine Ip80_ON(Irm)

Purpose:

To allocate one image reference memory to overlay graphics. This IRM becomes the default output for the IP80 Routines.

Parameter:

IRM - Integer*4 - Number in the range 0 to 20 that corresponds to the graphics output plane to be used.

Program Author: William Brent Lander, JAYCOR.

Date: March 10, 1983

8.14 Subroutine Ip80_ABox(Ix1, Iy1, Ix2, Iy2)

Purpose:

This subroutine puts up a rectangle of the default color in the overlay memory plane.

Parameters:

Ix1	- Integer -	Column number of the first column in the region to be colored.
Iy1	- Integer -	Row number of the first row in the region the region to be colored.
Ix2	- Integer -	Column number of the last column in the the region to be colored.
Iy2	- Integer -	Row number of the last row in the region the region to be colored.

Program Author: William Brent Lander, JAYCOR

Date: May 18, 1983

9. De Anza to Vax Image Transfers

The routines described in this section all deal with transferring data form the De Anza Image Refresh Memories (IRMs) to Fortran Byte arrays in the VAX. Routines are provided for transferring a row of pixels, a column of pixels, and a rectangular array of pixels. All transfers are between a single Fortran array and a single IRM of the De Anza.

Due to hardware constraints, it is not possible to transfer images both ways at the same time. In other words, once a call is made to IP8W_SETUP, no calls should be made to IP8R routines until the last line is sent with IP8W_SEND.

9.1 Subroutine IP8R_GETROW (Irm, ColNum, Array)

Purpose:

To transfer a single line of 512 bytes from one IRM of the De Anza.

Parameters:

Irm - Integer -	Number of the Image Refresh Memory in the IP8500 from which to transfer the image data. 0<=IRM<=20
ColNum - Integer -	Number of the column in the image data to obtain. 0 <=ColNum<= 511.
Array - Integer Array -	Array of at least 512 elements into which the image data is to be returned.

Program Author: William Brent Lander, JAYCOR.

Date: April 24, 1983

9.2 Subroutine IP8R_GETCOL (Irm, ColNum, Array)

Purpose:

To transfer a single column of 512 bytes from one IRM of the De Anza.

Parameters:

Irm - Integer - Number of the Image Refresh Memory in the IP8500 from which to transfer the image data. 0<=IRM<=20

ColNum - Integer - Number of the column in the image data to obtain. 0 <=ColNum<= 511.

Array - Integer Array - Array of at least 512 elements into which the image data is to be returned.

Program Author: William Brent Lander, JAYCOR.

Date: April 25, 1983.

9.3 Subroutine Ip8R_GetIrm (Irm, Ix1, Iy1, Ix2, Iy2, Array, M, N)

Purpose:

To transfer a rectangular portion of an IRM to an array in memory.

Parameters:

IRM - Integer - Number of the image refresh memory to transfer data from.

Ix1 - Integer - Column number of the first column in the region to be transferred. 0<=Ix1<=511

Iy1 Row number of the first row in the region to be transferred. 0<=Iy1<=511.

Ix1 - Integer - Column number of the last column in the region to be transferred. 0<=Ix2<=511

Iy1 Row number of the last row in the region to be transferred. 0<=Iy2<=511.

Program Author: William Brent Lander, JAYCOR.

Date: April 26, 1983.

10. Transferring Images from the VAX to the De Anza

The two routines described in this segment make it possible to transfer data to the De Anza one IRM at a time.

Due to hardware register problems, it is not possible at this time to do transfers to multiple IRM's at the same time, or to transfer information between the VAX and De Anza at the same time. For this reason, it is necessary to complete the data transfer initialized with a call to Ip8W_Setup before calling any Ip8R routine, or calling Ip8W_Setup again.

10.1 Subroutine Ip8W_Setup
(Irm, Ix1, Iy1, Ix2, Iy2, Array, M, N)

Purpose:

To transfer a rectangular portion of to IRM from an array in memory. This routine initializes the De Anza registers for a transfer of the requested mode. The actual data transfer is done in the program Ip8W_Send

Parameters:

```
        IRM - Integer - Number of the image refresh memory to
                 transfer data from.
        Ix1 - Integer - Column number of the first column in
                 the region to be transferred.  0<=Ix1<=511
        Iy1 - Integer - Row number of the first row in the
                 region to be transferred. 0<=Iy1<=511.
        Ix1 - Integer - Column number of the last column in
                 the region to be transferred.  0<=Ix2<=511
        Iy1 - Integer - Row number of the last row in the
                 region to be transferred. 0<=Iy2<=511.
        Imode - Integer - Output mode for data.
          Imode = 0 X-primary, Inc Primary, Inc Secondary
                = 1 X-primary, Dec Primary, Inc Secondary
                = 2 X-primary, Inc Primary, Dec Secondary
                = 3 X-primary, Dec Primary, Dec Secondary
                = 0 Y-primary, Inc Primary, Inc Secondary
                = 1 Y-primary, Dec Primary, Inc Secondary
                = 2 Y-primary, Inc Primary, Dec Secondary
                = 3 Y-primary, Dec Primary, Dec Secondary
```

Program Author: William Brent Lander, JAYCOR.

Date: July 8, 1983.

10.2 Subroutine Ip8W_Send(Array, Larray)

Purpose:

To transfer data to the De Anza from a vector of bytes in the VAX memory once the De Anza has been initialized by use of the Ip8W_Setup routine.

Parameters:

Array - Byte - Array of image data bytes. Each byte is assumed to be one pixel in size and in the range 0 to 255.
Larray - Integer - Number of bytes in the array to transfer.

Program Author: William Brent Lander, JAYCOR.

Date: July 8, 1983

11. Video Output Controller (VOC) Functions.

The importance of the Video Output Controller (VOC) can not be underestimated. It controls everything that happens to image or graphics data from the time it leaves the primary ITT (see the IP8I routines) up to the data display on the video monitor.

One of the most interesting features of the VOC is the file control registers/split screening. With split screening, it is possible to divide the screen into four logically discrete segments each of which is under the control of a file control register. The file control registers determine the source of input for video display in that segment, and final look-up table processing.

One application of the file control registers is to display a single color image (image only stored in one IRM) in multiple colors. In this way, the data display can be in black and white, or mapped into pseudo-color.

A second application is in enhancing figures consisting of a figure and ground (i.e. area or areas of interest, surrounded by areas of lesser importance). The graphics channel can be used to segment the display into a number of segments. Up to four different enhancement techniques can then be used on each of the logical segments.

11.1 Subroutine Ip8V_Voc(Voc)

Purpose:

To select the Video Output Controller (VOC) to be used in subsequent calls to IP8V routines. By default, VOC 0 is used.Once a selection is made, it stays in effect until a new selection is made.

Parameter:

VOC- Integer - Number (in range 0 to 3) of the video output controller to use for output.

Program Author: William Brent Lander, JAYCOR.

Date: January 8, 1983.

11.2 Subroutine Ip8V_AlphaMode(Mode)

Purpose:

To set the mode of the alphanumerics display.

Parameter:

Mode - Integer - Number in range 0 to 3. The effects are as follows:
0 - Alphanumerics pass through unmodified
1 - Alphanumerics color letters only, no matrix
2 - Alphanumerics are black only - no matrix
3 - Alphanumerics color goes to black, and black goes to full white.

Program Author: William Brent Lander,JAYCOR.

Date: 8 January 1984.

11.3 Subroutine Ip8V_SoftSplit

Purpose:

To turn on the software split screening mode of the VOC. In this mode, the high-order two bits of the graphics memory control which pixels are selected by which file control registers.

When called, the hardware split screen is disabled, and the contents of the hardware split screen position registers is disabled.

Parameters:

None.

Program Author: William Brent Lander, JAYCOR

Date: 8 January 1984.

11.4 Subroutine Ip8V_Hardsplit(Xline, Yline)

Purpose:

This routine divides the screen into four regions based on display line number. Each region is under the control of a separate file control register. The file control registers (and their associated look-up tables) can be used to alter the video displayed in this region. Regions are labeled thus:

```
(0,511)+-------+-------+(511,511)
       !       !       !
       !   1   !   0   !
       !       !       !
       +-------+-------+(511,Yline)
       !       !       !
       !   2   !   3   !
       !       !       !
  (0,0)+-------+-------+(511,0)
           (Xline,0)
```

Numbers in parenthesis are line numbers. Numbers in boxes are region numbers.

Parameters:

Xline - Integer - Number in range -1 to 511 containing the value to be used for vertical split screen division. If -1, no change is made to the register from the previous value.
Yline - Integer - Number in range -1 to 511 containing the value to be used for horizontal split screen division. If -1, no change is made to the register from the previous value.

Program Author: William Brent Lander, JAYCOR.

Date: January 8, 1983.

11.5 Subroutine Ip8V_ReadSplit(Mode, Xline, Yline)

Purpose:

To obtain the current values used in split screening operations. This routine does not alter the value of any registers in the DeAnza.

Parameters:

Mode - Integer - Returns an indicator of the type of split screening operations are in effect. 0-Software split, 1-Hardware
Xline - Integer - Number in range 0 to 511 containing the value being used for vertical split screen division.

Yline - Integer - Number in range 0 to 511 containing the value being used for horizontal split screen division.

Program Author: William Brent Lander, JAYCOR.

Date: January 8, 1983.

11.6 Subroutine Ip8V_Define (SegmentNo, OutputChannel, IRM, LUT)

Purpose:

To define the translation to be applied to image output going through the default VOC.

Parameters:

SegmentNo - Integer - Number in the range 0 to 3which is the number of the region to be altered by this instruction.

OutputChannel - Integer - Number in the range 0 to 3 which is the number of the video output channel on the back of the DeAnza/Gould device hooked to the monitor. Normally 0-red, 1-green, 2-blue, 3-b/w

IRM - Integer - Number of the image refresh memory to be connected to this region and channel. Only one IRM may be connected to a region/outputchannel pair. IRM must be in the range of 0 to 20.

LUT - Integer - Number in the range -1 to 3. If negative no look-up table processing is done before the data is displayed,otherwise LUT is the number of the look-up table. There are four look-up tables for each display segment. The look-up tables can be loaded via IP8V_LOAD, IP8V_RAMP, or IP8V_THRESHOLD.

Program Author: William Brent Lander, JAYCOR.

Date: January 8, 1983.

11.7 Subroutine Ip8V_Describe (SegmentNo, OutputChannel, IRM, LUT)

Purpose:

To describe the translation being applied to image output going through the default VOC.

Parameters:

SegmentNo - Integer - Number in the range 0 to 3. The number of the region to be described.
OutputChannel - Integer - Number in the range 0 to 3. The number of the video output channel on the back of the DeAnza/Gould device hooked to the monitor. Normally 0-red, 1-green, 2-blue, 3-b/w
IRM - Integer - Returns the number of the image refresh memory connected to this region and channel. IRM will be in the range 0 to 20.
LUT - Integer - Returns a number in the range -1 to 3. If negative no look-up table processing is done before the data is displayed, otherwise LUT is the number of the look-up table. There are four look-up tables for each region/output channel pair. The look-up tables can be read using IP8V_READ.

Program Author: William Brent Lander, JAYCOR.

Date: January 8, 1983.

11.8 Subroutine Ip8V_Threshold (OutputChannel, Lut, Min, Max)

Purpose:

To allow for the Intensity transformation arrays in the VOC to be loaded from a program with a ramp function, or portion of a ramp function.

Parameters:

OutputChannel - Integer - Number in the range 0 to 3 specifying the number of the output channel in which the LUT is to be applied. Usually, 0-red, 1-green, 2-blue, 3-b/w
LUT - Integer - Number in the range 0 to 3. Numbers specified by JTABLE are loaded into the corresponding look-up table for the default memory controller
Min - Integer - First input intensity to be assigned a value

Max - Integer - Last input intensity to be assigned a value

Program Author: William Brent Lander, JAYCOR.

Date: 8 January 1984

Note:

With this subroutine, the value assigned in the ramp corresponds to the unmodified intensity.

11.9 Subroutine Ip8V_Load (OutputChannel, Lut, Jtable)

Purpose:

To load a selected set of image refresh memories with a user-supplied Intensity transformation unit (ITU)

Parameters:

OutputChannel - Integer - Number in the range 0 to 3 specifying the number of the output channel in which the LUT is to be applied. Usually, 0-red, 1-green, 2-blue, 3-b/w

LUT - Integer - Number in the range 0 to 3. Numbers specified by JTABLE are loaded into the corresponding look-up table for the default memory controller.

JTable - Integer array of 256 elements - The user supplied values to be sent to the LUT's

Program Author: William Brent Lander, Jaycor.

Date: 8 January 1984.

11.10 Subroutine Ip8V_Ramp (OutputChannel, Lut, Min, Max, Intenmin, Intenmax)

Purpose:

To allow for the Intensity transformation planes to be loaded from a program with a ramp function, or portion of a ramp function.

Parameters:

OutputChannel - Integer - Number in the range 0 to 3 specifying the number of the output channel in which the LUT is to be applied. Usually, 0-red, 1-green, 2-blue, 3-b/w

LUT - Integer - Number in the range 0 to 3. Numbers specified by JTABLE are loaded into the corresponding look-up table for the default memory controller.

Min - Integer - First input intensity to be ascribed a value

Max - Last input intensity to be ascribed a value

Program Author: William Brent Lander, Jaycor.

Date: 8 January 1984.

11.11 Subroutine Ip8V_Read (Intensity, ChannelNo, Itu)

Purpose:

To allow for the contents of the intensity transformation table (ITU) to be recovered at any time.

Parameters:

Intensity - Integer*4 - array returning the values stored in the requested LUT.

ChannelNo - Integer*4 - The number of the output channel associated with the LUT. Value should be in the range 0 to 3.

Lut - Integer*4 Number of the intensity look-up table to read. Value should be in the range 0 to 3.

Program Author: William Brent Lander, Jaycor.

Date: 8 January 1984.

MATH

Image Manipulation Program

Math Image Transfer Program

By

William Brent Lander

JAYCOR

MATH - Image Transfer and Calculation Program

1. Introduction

MATH is a program to do image to image calculations via a simple algebraic annotation, and gather basic statistics on image operations. There are two forms for commands.

Form 1:

```
math> DESTINATION = EXPRESSION
```

Form 2:

```
math> COMMAND EXPRESSION
```

Where COMMAND is one of the verbs discussed in section 3.

1.1 MATH Vita

MATH was developed at the Naval Research Laboratory Code 6520 Advanced Concepts Branch Computer Facility. The computer code was written and debugged on the VAX 11/780 computer with the De Anza/Gould IP8500 image processor. FORTRAN Four Plus, as provided by Digital Equipment Corporation, was used as the host language, although many of the operating system services were used. The program was developed under versions 3.2, 3.3, and 3.4 of VMS. The Level 0 (Device Driver) of the LIPS 3.0 package was used. All other image processor control functions are done through custom software.

The 3400 lines of code was designed and implemented by William Brent Lander. Debugging has been done while the program was in use for several months at NRL.

2. Form 1 Commands: Arithmetic Assignment

The Form 1 commands are the active commands to do image transfer. The basic form of the MATH assignment statement is as follows:

```
math> {destination} = {expression}
```

The following subsections describe the portions of the math command.

2.1 DESTINATION

{destinations} can be any of the following:

1) A number from 0 to 20. This will cause the output to go to that Image Refresh Memory (IRM)

2) Filename. This will cause the output to go to the requested file. New files are created in standard image format.

2.2 EXPRESSION

An expression can be any of the following:

1) A filename. The contents of the file (in standard format) is copied to the output. The only restriction on file names is, the minus symbol cannot appear in the file name, such as in relative directory addressing (i.e. [-.SUB2]PICTURE.DAT is not acceptable)

2) A numeric constant. Sets the output to the given value. This can be useful in clearing a IRM.

3) An alphanumerics string of the form IRMn where n is an integer in the range 0 to 20. This causes the input to be from a given IRM.

4) An equation, in infix notation, which may combine items from 1, 2, or 3. Parenthesis can be used to specify order of evaluation.

The following is a list of examples of various expressions.

1) Simple Filename:

```
PICTURE.B1
```

2) Add a file to image memory zero

```
PICTURE.B1+IRM0
```

3) Compound image operations

```
90*LOG(PICTURE.B1/PICTURE.B2)
```

2.2.1 OPERATORS

The valid algebraic operators are defined as follows:

```
+ Add two values
- Subtract two values
* Multiply two values
/ Divide two values
> Returns the value of the left if greater than
          the value on the right, otherwise 0.
< Returns the value of the left if less than the
          value on the right, otherwise 0.
& Logical (Bitwise) AND operation on two values
| Logical (Bitwise) OR  operation on two values
```

(...) Parentheses can be used anywhere logically significant.

Operators have the same presidence as in Fortran. If there is no corresponding operator, the operator has the same presidence as the addition operator (+).

The operators can be used between any two valid simple expressions or functions, or combined in any arbitrary fashion to any any depth. Operators can be used to combine simple expressions into one argument for functional evaluation.

The following are some valid expressions:

```
PIC.DAT+10
100*(Pic.Dat+25)/Pic2.Dat
((PIC.DAT/(3.19*LOG(PIC.DAT))
```

The following are invalid expressions:

```
+ 10 PIC.DAT   ........ No first argument to add
PIC . DAT      ........ Blanks in file name
(A.PIC +B.Pic  ........ Parenthesis don't balance
```

2.2.2 FUNCTIONS

A number of functions are available in MATH expressions:

```
ABS(argument) ....Returns the absolute value of
                        the argument
COS(argument) ....Returns the trigonometric
                        COSINE of the argument
EXP(argument) ....Returns the argument raised to
                        the power of e.
SIN(argument) ....Returns the trigonometric SINE
                        of the argument
SQRT(argument) ...Returns the positive square
                        root of the argument
TAN(argument) ....Returns the trigonometric
                        TANGENT of the argument
LOG(argument) ....Returns the logarithm base e of
                        the argument
```

(...) Parentheses must be used around the argument.

3. Form 2 Commands

The following subsections describe the form 2 commands of the MATH program. Each command is described in its own segment. Headings under each command describe the possible parameters.

Form 2 commands deal with two groups of commands. One group of commands is used to control the output of images or analysis. The second group deals with simple image measurements.

3.1 COUNT

```
math> COUNT expression
```

Causes the expression to be evaluated and a count of the number of times the expression was zero, and not zero will be printed. No change to the displayed image is produced.

3.2 END

```
math>END
```

Will cause the math program to terminate. MATH can also be exited by entering EXIT, or Control-Z.

3.3 HELP

MATH has an on-line help facility that can be used interactively to obtain information about the MATH commands. Approximately eight hundred lines of documentation are available.

To use the help, type HELP in response to the MATH prompt:

```
math> HELP
```

A brief introduction to MATH will be typed, followed by a listing of available subtopics. By typing the name of a subtopic, you can progress to the next level of help. By typing in a blank line, you can move up one level. By typing in Control-Z, you can return to the main level MATH prompt.

3.4 HISTOGRAM

```
math> HISTOGRAM expression
```

Causes the expression to be evaluated and a histogram of the number of times each intensity occurred will be printed out. No change to the displayed image is produced.

3.5 OUTPUT

```
math>       OUTPUT <ascending|descending>
                   <flipped|notflipped>
                   <vertical|horizontal>
                   <?>
```

Controls the method of storage of the output. Ascending|Descending and Flipped|Notflipped apply to both file and image output. Vertical|Horizontal applies only to image output and not to the file output. The "?" will cause the current setting to be displayed.
For example, to specify that the output should be across the screen from left to right with the image line-wise reversed, the following command would be used:

```
filter> Output Horizontal Flipped
```

3.5.1 ASCENDING

Causes the output to start with the first record in the file/ line 0 of the display, and output to sequentially higher numbered records.

3.5.2 DESCENDING

Causes the output to start with the last record in the file/line 511 of the display, and output to sequentially lower numbered records.

3.5.3 FLIPPED

Causes the output to be transposed on a line by line basis before output. The first pixel is exchanged with the last pixel on the line, the second pixel is exchanged with the next to last pixel, and so on.

3.5.4 HORIZONTAL

Causes the output to be painted to the screen from left to right, or right to left, as opposed to top to bottom.

3.5.5 NOTFLIPPED

Causes the output to be in the same record for record order as input. The first pixel in the image file is the first pixel in the output.

3.5.6 VERTICAL

Causes the output to be painted to the screen from top to bottom, or bottom to top.

3.6 PRINT

The print command is the method by which output of the basic image measurements can be sent to either a file or output device such as a line printer. If output is sent to the line printer, the output from all instances of the PRINT command are stored and printed when the program is terminated.

3.6.1 ON

```
math> PRINT ON <filename or device>
```

Causes all further output of PRINT commands to be sent to the specified filename or device. By default, output is sent to LPA0:.

3.6.2 COMMAND

```
math> PRINT <COUNT .....>
            <HISTOGRAM..>
            <STATS .....>
```

Causes the requested command to be executed and the generated report to be sent to the specified output device. For example, to print the statistics on IRM 0, the command would be:

```
math>PRINT STATS 0
```

3.6.3 NOW

```
math> PRINT NOW
```

By default, the reports to be printed are stored up and printed when MATH is exited. In some cases, it could be necessary to obtain the report as soon as possible. In that case, entering the PRINT NOW command will cause the output to be queued for processing at this time.

3.7 STATS

```
math> STATS n
```

Causes the program to print the minimum, maximum, average, and standard deviation of the intensity values in the image displayed in IRM n. If n is not specified, it will cause the computer to attempt to output the statistics for IRM's 0 through 3, inclusive.

3.8 $ - Executing a DCL Command

```
math> $<DCL-COMMAND>
```

Causes the MATH program to "go to sleep" and runs the requested DCL command. Very useful in cases where the name of the picture file has been forgotten and a directory command would be useful.

3.9 @ - Command_File

```
math>@<MATH-COMMAND-FILE>
```

Causes the MATH program to run with input from a file, as opposed to the terminal. Useful when the same set of operations will be used over and over. Any math command is valid in a command file.

3.10 ! - Comment

If the first non-blank symbol is an exclamation mark(!) the line is ignored. Comment lines are especially useful either in command files or in producing examples of terminal sessions.

Example:

```
        math> !The following code shows a technique
        math> !for getting plant water measurements from
        math> ! Landsat-D images
        math> !
        math> O=100*PIC.B1/PIC.B6
        math> ! . . .
```

4. EXAMPLES

1) Clear graphics overlay (Channel 3) memory.

```
                    $ MATH 3=0
```

2) Put up an image from file XYZ.PIC into memory channel 0 (RED) and doubling the intensity.

```
                    $ MATH
                    math>0=XYZ.PIC*2
                    math>^Z
```

3) Double the intensity of a color picture already stored in image planes 0, 1, and 2

```
                    $ MATH
                    math>0=IRM0*2
                    math>1=IRM1*2
                    math>2=IRM2*2
                    math>^Z
```

4) Find out the average, minimum, maximum, and s.d. of the image that is currently displayed in IRM 0, but was not displayed by the MATH program

```
          $ MATH
          math>0=irm0
          math>stats 0
          Statistics for IRM 0
          Number of points: 262144
          Min:XXX                     Average:XXXXXXX
          Max:XXX                     Standard Dev:XXX

          math>^Z
          $ . . .
```

5) Print the histogram of the average value of two pictures

```
          $MATH
          math>PRINT HISTOGRAM (PIC1.B1+PIC2.B1)/2
          math>PRINT NOW
          math> . . .
```

5. Notes

1) Results of operations are truncated to the range 0 to 255 before output.

2) No uniary operators, other than functions, are allowed.

3) All constants are positive.

4) Minus signs [-] are not allowed as part of file names.

5) Blanks between filenames, operators, constants, IRM specifiers are optional, but cannot be imbedded in the tokens themselves.

6) After a command has been entered, it may take a significant amount of time (up to 2 minutes) before the display is complete. If the command to display the data is in error, this can result in an annoying amount of time lost. If it is desirable to terminate a command before completion, the user can type <Control-C> to break the execution of the command, and return control to the next higher level. Striking <Control-C> twice in a row, or striking <Control-Y> will terminate the program.

7) Standard Image Files are defined to be fixed length 512 byte, unformatted files, 512 records long. The file must be able to be opened direct access.